Introduzione a Scilab

Politecnico di Milano
Sede di Cremona

Giovanni Vannozzi

A.A. 2010-2011

A cosa serve questa presentazione

Lo scopo di questo materiale è quello di fornire le **informazioni necessarie** per un **adeguato utilizzo** del **programma**

Scilab

Scientif Laboratory

e del corrispondente **ambiente** afferente tutte le **procedure** per la **simulazione**

Scicos

Scilab Connected Object Simulator

Indice del materiale didattico
Finalità ed Obiettivi

- ⌘ **Descrizione generale di Scilab**
- ⌘ **Alcune funzioni predefinite**
- ⌘ **Definizione di matrici e vettori**.
- ⌘ **Definizione di polinomi**.
- ⌘ Rappresentazione dei **sistemi dinamici lineari**
- ⌘ **Rappresentazione grafica dei dati**.
- ⌘ **L’ambiente di simulazione Scicos**.

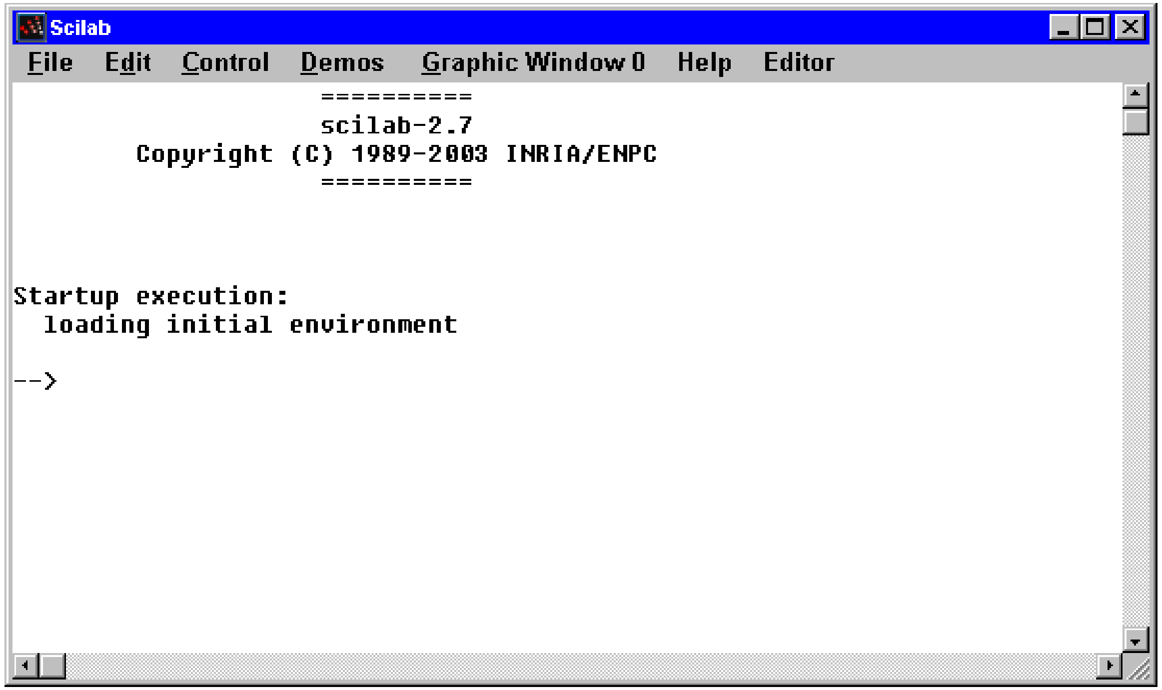
Descrizione generale di Scilab

SCILAB (**Scientific Laboratory**):

- **linguaggio di programmazione per le varie applicazioni scientifiche e numeriche**;
- **set di funzioni predefinite**;
- **interprete di comandi**;
- **possibilità di scrivere nuove funzioni e di importare moduli di codice Fortran o C**;
- **libreria con toolbox** per applicazioni specifiche (***sistemi di controllo lineari e non lineari, analisi dei segnali, analisi nonché sintesi di controllori***,…).

L'interfaccia di Scilab

- **Interfaccia utente**: la **Scilab Window** dà accesso diretto **all'interprete** (ovvero, alla **scrittura diretta dei comandi**)



Scilab come calcolatrice...

- **La modalità di impiego più semplice è la valutazione di espressioni numeriche.**
- Esempio: per **calcolare:** $4+\sqrt{2}-\sin(0.2\pi)^2+e^2$

  è **sufficiente digitare** al **prompt**:

  -->**4 + sqrt(2) - sin(0.2*%pi)^2 + %e^2**

  ans =

  12.457778
- Il **risultato** viene **memorizzato** nella **variabile standard *ans***.

Le variabili predefinite

- Esiste un discreto numero di **variabili** che sono già **predefinite** per **default** da **Scilab**; alcune di queste **variabili predefinite** sono:

%i	= **unità immaginaria:**	**$i^2=-1$ ovvero $i=\text{sqrt}(-1)$**
%e	= **numero di Nepero:**	**e=2.7182818**
%pi	= **numero pi-greco:**	**π =3,1415927**
%eps	= **precisione macchina:**	**eps=2,220 10^{-16}**
%T	= **variabile logica TRUE**	**(T)**
%F	= **variabile logica FALSE**	**(F)**
%inf	= **Inf infinito**	**(simbolo ∞)**
%nan	= **Nan**	**(Not a number)**

- Per **cancellare** tutte le **variabile definite** dallo utente si utilizza il comando:

  --> **clear**

Definizione delle variabili

- E' possibile definire **variabili** ed espressioni non numeriche più complesse.
- Esempio 1:

```
--> a=5; b=3;c=4;
--> a*b/c
 ans =
    3.75
```

Esempio 2:

```
--> a=4; b=2;c=5;
--> (a/b).^c
 ans =
    32.
```

- Per **cancellare** una variabile definita dallo utente (per esempio, **a**) si usa il comando:

```
--> clear a
```

Variabili definite in memoria

- Ogni variabile definita tramite assegnazione viene conservata in **memoria**.
- Il comando ***whos()*** mostra una **lista** delle **variabili** definite con assegnazione:

*-->**whos()***

Name	*Type*	*Size*	*Bytes*
ans	*constant*	*1 by 1*	*24*
b	*constant*	*1 by 1*	*24*
a	*constant*	*1 by 1*	*24*
c	*constant*	*1 by 1*	*24*

LE FUNZIONI PREDEFINITE

- Esiste un **insieme** molto vasto di **funzioni predefinite**, come ***sin*** e ***sqrt*** nell'esempio precedente.
- Un'altra funzione predefinita utile nel calcolo combinatorio è ***factorial*** che **restituisce** il **valore** del **fattoriale** del **numero intero n**:
  n! = factorial(n)
  --> fatt = factorial(7)   fatt = 5040
- A **differenza** dei normali linguaggi (Pascal, C,..) **NON occorre dichiarare le variabili**. L'assegnazione coincide con la dichiarazione

Esempi di funzioni predefinite
(di uso più comune)

- **Funzioni trigonometriche** (***sin***, ***cos***, ***tan***, ***acos***, ***asin***, ***atan***, ...);
- **Esponenziale e logaritmo** (***exp***, ***expm***, ***log***, ***log2***, ***log10***, ***log1p***, ***logm***, ...);
- **Altre funzioni elementari** (***sqrt***, ***max***, ***min***, ***rand***, ***sign***, ***size***, ...);
- **Numeri complessi** (***abs*** → modulo, ***phasemag*** → modulo e fase, ***real*** → parte reale, ***imag*** → parte immaginaria).

Una funzione fondamentale

help

- ***help*** **seguito dal nome della funzione** restituisce, attraverso la guida in linea, una **descrizione** e la **sintassi** d'uso della stessa **funzione**;

- ***help***, **senza ulteriori specifiche**, restituisce **l'indice degli argomenti** della **guida in linea** di **Scilab**.

Definizione numeri complessi

- In Scilab un **numero complesso** è così definito:

Esempio: **definire** il **numero complesso z=3+4i**

```
-->z=3+4*%i          oppure     -->i=%i;z=3+4*i
 z =                             z =
      3. + 4.i                        3. + 4.i
 -->a=real(z)        oppure      -->a=real(3+4*i)
 a =                             a =
      3                               3
-->b=imag(z)         oppure     -->b=imag(3+4*i)
 b =                             b =
      4                               4
```

Numeri complessi (2)

⌘ In **Scilab** il **modulo** e la **fase** di un **numero complesso** si determinano con i **due** comandi:

Esempio: **definito il numero complesso: z=3+4i**

```
-->rho=abs(z)        oppure      -->rho=abs(3+4*i)
 rho  =                           rho  =
      5                                5
-->theta=phasemag(z)
 theta  =
          53.130102
```

53.130102 (**angolo espresso in gradi**)

In **Scilab** il **coniugato** del **numero complesso** **z=3+4i** è fornito dalla **funzione**: **conj(z)**.

--> **w = conj(z)** fornisce **w = 3-4i**

Numeri complessi (3)

- La funzione **phasemag** di **Scilab** ammette un secondo utilizzo tramite una specifica **sintassi** in base alla quale si richiede la restituzione dei **valori** del **modulo** e della **fase** di un **numero complesso.**

Esempio: **definito il numero complesso: z=3+4i**

```
-->[theta rhodb]=phasemag(z)
 rhodb  =
        13.9794
 theta  =
        53.130102
```

N.B.: L'angolo viene espresso in gradi e il modulo in decibel

Numeri complessi (4)

- **Scilab** consente di eseguire le normali operazioni di **somma**, **differenza**, **prodotto, quoziente** e **potenza** e **radice** di **numeri complessi**.

Esempio 1: somma e differenza

```
-->z1=4-3*%i; z2=6+8*%i;
-->zs=z1+z2          -->zd=z1-z2
 zs =                 zd =
     10. + 5.i            - 2. - 11.i
```

Esempio 2: prodotto e quoziente

```
-->zpro=z1*z2        -->zquo=z2/z1
 zpro =               zquo=
       48. + 14.i           2.i
```

Numeri complessi (5)

Esempio 3: elevamento a potenza

```
-->z1=4-3*%i;z2=6+8*%i;
```

```
-->zp1=z1^3
 zp1  =
     -44. - 117.i
```

```
-->zp2=z2^2
 zp2  =
     - 28. + 96.i
```

Esempio 4: estrazione di radice

```
-->zr1=z1^(1/2)
 zr1  =
     2.1213203 - 0.7071068i
```

N.B.: Viene fornita la radice principale.

Definizione di matrici (1)

- Come si definisce una **matrice** in **Scilab**?

  Esempio: **definire** la **matrice 2x2**

```
-->A=[1,2;3,4]
 A  =

    !   1.    2. !
    !   3.    4. !
```

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

- **Come si accede agli elementi di una matrice**?

```
--> A(1,2)
 ans  =

       2
```

Indici di riga e di colonna dell'elemento di interesse

Definizione di matrici (2)

- Costruzione di matrici da matrici assegnate:

Esempio: **assegnate le matrici**:

```
-->A=[1,2,3;2,5,1];
-->B=[2,1,4];
```

$$A=\begin{pmatrix}1 & 2 & 3\\ 2 & 5 & 1\end{pmatrix}$$

$$B=\begin{pmatrix}2 & 1 & 4\end{pmatrix}$$

si ottiene, per costruzione, la **matrice C** con:

```
--> C=[A;B]
 C =
       ! 1.  2.  3. !
       ! 2.  5.  1. !
       ! 2.  1.  4. !
```

$$C=\begin{pmatrix}1 & 2 & 3\\ 2 & 5 & 1\\ 2 & 1 & 4\end{pmatrix}$$

Definizione di matrici (3)

- Costruzione di matrici da matrici assegnate:

Esempio: assegnate le matrici:

```
-->A=[1,2,3;2,5,1];
-->B=[2,1,4];
-->D=[1;7;9];
```

$$D = \begin{pmatrix} 1 \\ 7 \\ 9 \end{pmatrix} \quad A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 2 & 1 & 4 \end{pmatrix}$$

si ottiene, per costruzione, la **matrice E** con:

```
--> E=[[A;B] D]      oppure: E=[[A;B],D]
 E =

      ! 1.  2.  3.  1. !
      ! 2.  5.  1.  7. !
      ! 2.  1.  4.  9. !
```

$$E = \begin{pmatrix} 1 & 2 & 3 & 1 \\ 2 & 5 & 1 & 7 \\ 2 & 1 & 4 & 9 \end{pmatrix}$$

Definizione di matrici (4)

- **Costruzione di matrici** da matrici assegnate:

Esempio: **assegnate le matrici**:

```
-->F=[1 2;3 4];
-->G=[5,6];
-->H=[7 8 9]
```

$$F=\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

$$G=\begin{pmatrix} 5 & 6 \end{pmatrix}$$

$$H=\begin{pmatrix} 7 & 8 & 9 \end{pmatrix}$$

si ottiene, per costruzione, la **matrice L** con:

L=[[F,G'];H] oppure: **L=[F,G';H]**

```
L =

      | 1.  2.  5. |
      | 3.  4.  6. |
      | 7.  8.  9. |
```

$$L=\begin{pmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

MATRICI PARTICOLARI (1)

- **La matrice UNITÀ:**

  Esempio: **assegnamento matrice unità di ordine 3:**

```
-->A=eye(3,3)
 A =
      | 1.  0.  0. |
      | 0.  1.  0. |
      | 0.  0.  1. |
```

```
-->U=eye(3,5)
 A =
      | 1.  0.  0.  0.  0.|
      | 0.  1.  0.  0.  0.|
      | 0.  0.  1.  0.  0.|
```

- **Le matrici ones e zeros**

```
--> B=ones(3,3)
 B =
      | 1.  1.  1. |
      | 1.  1.  1. |
      | 1.  1.  1. |
```

```
--> C =zeros(3,4)
 C =
      | 0.  0.  0.  0. |
      | 0.  0.  0.  0. |
      | 0.  0.  0.  0. |
```

MATRICI PARTICOLARI (2)

- Assegnata la **matrice A**

  --> A=[1 2;3 4]

  le seguenti matrici notevoli si ottengono con i comandi:

- **Matrice trasposta**:

  --> **A'**

  ans =

  ```
  !   1.0   3.0 !
  !   2.0   4.0 !
  ```

- **Matrice inversa**:

  --> **inv(A)**

  ans =

  ```
  !  -2.0    1.0 !
  !   1.5   -0.5 !
  ```

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

$$A' = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

$$A^{-1} = \begin{pmatrix} -2 & 1 \\ 1{,}5 & -0{,}5 \end{pmatrix}$$

Colon symbol (:)

- Per accedere a **intere righe o colonne** di una **matrice**, si usa il carattere ***colon symbol :***

- Esempio: **selezionare la prima riga di A**

  --> **A(1,:)**

  ans =

  ! **1.** **2.** !

$$ans = \begin{bmatrix} 1 & 2 \end{bmatrix} \leftarrow A = \begin{bmatrix} \boxed{1 \quad 2} \\ 3 \quad 4 \end{bmatrix}$$

- Esempio: **selezionare la seconda colonna di A**

  --> **A(:,2)**

  ans =

  ! **2.** !

  ! **4.** !

$$ans = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \leftarrow A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Selezionare sottomatrici (1)

- Definendo:

```
--> B=[1,2,3;4,5,6]
 B =
!   1.   2.   3. !
!   4.   5.   6. !
```

- Si ottiene, con l’istruzione seguente:

```
--> B(1:2,2:3)
 ans =
!   2.   3. !
!   5.   6. !
```

Indici della sottomatrice di interesse

Selezionare sottomatrici (2)

- Definendo la **matrice B**:

--> **B=[1,2,3;4,5,6;7,8,9]**

```
B =
!  1.   2.   3. !
!  4.   5.   6. !
!  7.   8.   9. !
```

$$B = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \rightarrow A = \begin{pmatrix} 2 & 3 \\ 8 & 9 \end{pmatrix}$$

- Si ottiene la **sottomatrice A** mediante:

--> **A= B([1 3],[2 3])**

```
A =
!  2.   3. !
!  8.   9. !
```

Indici delle righe e delle colonne della sottomatrice di interesse

Operazioni elementari sulle matrici (1)

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

- **Determinante**:

  --> **det(A)**

  ans =

  - 2.

Rango:

--> **rank(A)**

ans =

2.

- **Triangolazione**

  --> **triu(A)**

  ans =

  | **1. 2.** |
  | **0. 4.** |

  --> **tril(A)**

  ans =

  | **1. 0.** |
  | **3. 4.** |

Matrice diagonale

--> **D= diag([2;5])**

D =

| **2.   0.** |
| **0.   5.** |

Operazioni elementari sulle matrici: traccia, rango ..... (2)

- **traccia**:
  --> **trace(K)**
  ans =
  14.

$$K = \begin{bmatrix} 1 & 3 & 2 \\ 2 & 4 & 7 \\ 1 & 5 & 9 \end{bmatrix}$$

diagonale
--> **diag(K)**
ans =
1.
4.
9.

dimensioni di una matrice:
--> **size(K)**
ans =
3. 3.

--> **[r c] = size(K)**
c =
3
r =
3

Operazioni elementari sulle matrici: prodotto righe per colonne (3)

- Sono definiti gli **operatori**: **+**, **-**, *****, **.***, **^** e **.^**
- **Prodotto righe per colonne:** (**prodotto fra matrici conformabili**)

```
--> A=[2 3 1;1 5 2;3 3 1];
--> B=[1 1 3;2 1 5;3 1 2];
--> PRC=A*B      *= prodotto righe per colonne
 PRC =
        | 11.   6.   23. |
        | 17.   8.   32. |
        | 12.   7.   26. |
```

Operazioni elementari sulle matrici: prodotto scalare (4)

⌘ **Prodotto scalare:**
(**prodotto elemento per elemento**)

```
--> A=[2 3 1 1;1 5 2 2;3 3 1 2];
--> B=[1 1 3 2;2 1 5 3;3 1 2 5];
--> PS=A.*B
```

.* = prodotto scalare

```
PS =
   | 2.   3.    3.    2. |
   | 2.   5.   10.    6. |
   | 9.   3.    2.   10. |
```

Operazioni elementari sulle matrici: potenza di una matrice (5)

- **Elevamento a Potenza:**
  (**potenze di matrici quadrate**)

```
--> A=[2 3 1;1 5 2;3 3 1];
--> PT2=A^2
```

^= **elevamento a potenza**

PT2 =

$$\begin{vmatrix} 10. & 24. & 9. \\ 13. & 34. & 13. \\ 12. & 27. & 10. \end{vmatrix}$$

(Equivale all'operazione di **prodotto**: **PT2 = A*A**)

Operazioni elementari sulle matrici: potenza di singolo elemento (6)

- **Elevamento a Potenza (scalare):** (**potenza elemento per elemento**)

--> A=[2 3 1;1 2 4;3 3 5];

--> PT3=A.^3   .^ = elevamento a potenza

PT3 =

$$\begin{vmatrix} 8. & 27. & 1. \\ 1. & 8. & 64. \\ 27. & 27. & 125. \end{vmatrix}$$

(equivale al **prodotto**: **PT3 = A.*A.*A**)

Operazioni elementari sulle matrici – Diagonalizzazione (7)

- **Assegnata la matrice A, gli autovalori si determinano con le istruzioni seguenti:**

```
--> A=[1 2 2;3 0 2;1 1 3];
--> lambda = spec(A)
 lambda =
        | -2. |
        |  5. |
        |  1. |
```

viene restituito il vettore colonna costituito dagli **autovalori** della **matrice A**: $\det(A-\lambda I)=0$

Operazioni elementari sulle matrici – Diagonalizzazione (8)

⌘ Assegnata la matrice A con il comando:

--> A=[1 2 2;3 0 2;1 1 3];

l'istruzione seguente:

--> [P D] = spec(A)

fornisce:

- la matrice diagonale D avente sulla diagonale principale gli autovalori della matrice A;
- la matrice di passaggio P le cui colonne sono costituite dagli autovettori associati ai rispettivi autovalori. Gli autovettori sono normalizzati.

Operazioni elementari sulle matrici – Diagonalizzazione (9)

- L'istruzione **spec(A)** fornisce, pertanto:

  --> **[P D] = spec(A)**

```
D =  ! -2.  0.  0. !
     !  0.  1.  0. !
     !  0.  0.  5. !
```

$$P^{-1} * A * P = D$$

```
P =  !  0.5230  -0.5774   0.5774 !
     ! -0.8498  -0.5774   0.5774 !
     !  0.0654   0.5774   0.5574 !
```

Definizione dei vettori (1)

- --> **v=(0:10)** equivale a --> **v=(0:1:10)**

  v =

  ! 0. 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. !

- --> **v=(1:0.5:3)**

  v =

  ! 1. 1.5 2. 2.5 3. !

Valore iniziale   Passo   Valore finale

Definizione di vettori (2)

- È possibile **definire** un **vettore** le cui **componenti** soddisfano **differenti tipi** di **progressioni** o **norma costitutiva**:

```
--> Vet=[1:3:10 15:6:27]
Vet =
 ! 1.   4.   7.   10.   15.   21.   27. !
```

1° valore iniziale

1° Passo

1° valore finale

2° valore iniziale

2° Passo

2° valore finale

Definizione di vettori (3)

- **Vettori come matrici riga**:

  --> **vr=[3 6 1]** **oppure** --> **vr=[3;6;1]'**

  vr =

  ! **3.** **6.** **1.** !

$$v_r = v_c' \qquad v_c = v_r'$$

- **Vettori come matrici colonna**:

  --> **vc=[3; 6; 1]** **oppure** --> **vc=[3 6 1]'**

  vc =

  ! **3.** !
  ! **6.** !
  ! **1.** !

Definizione di particolari vettori

- Vettori riga con componenti linearmente equidistanti fra loro: suddivisione di un intervallo in n parti uguali.

```
--> VL=linspace(1,10,5)
 VL =
        ! 1.   3.25   5.5.   7.75   10. !
```

- Vettori riga con componenti corrispondenti alla suddivisione di un intervallo in n parti uguali su scala logaritmica a base 10.

```
--> VC=logspace(0,1,5)
 VC = ! 1.  1.778279  3.162277  5.623413  10. !
```

OPERAZIONI CON VETTORI (1)

- Sono definiti gli **operatori +,-,*,.*** e **.^**

-->> **v1=[1 3 2 4 2]**

-->> **ve=v1.^3** (**potenza di ogni singolo elemento**)

ve =

! **1. 27. 8. 64. 8.** !

-->> **v1=[1 3 2 4 2] ; v2=[3 5 4 1 6];**

-->> **vp=v1.*v2** (**prodotto elemento per elemento**)

vp =

! **3. 15. 8. 4. 12.** !

L'operazione di prodotto scalare è commutabile.

OPERAZIONI CON VETTORI (2)

- I **vettori riga** e **colonna** sono, nella sostanza, delle **particolari matrici rettangolari.**
- Pertanto, è definito anche per i vettori il prodotto inteso come **prodotto di matrici conformabili.**
- **Esempio**: assegnati i seguenti **vettori**:

 --> **v1=[1 3 2 4 2] ; v2=[3 5 4 1 6];**

 l'operazione di **prodotto fra vettori**, espressa come **prodotto di matrici**, è definita da *****:

--> **vpm=v1*v2'** (**prodotto di v1 per il trasposto di v2**)

 vpm =

 42 ⟷ **(1·3 + 3·5 + 2·4 + 4·1 +2·6)**

La Definizione dei POLINOMI

- Esempio: si vuole definire il polinomio: **s^2-5s+6**

Definizione mediante i coefficienti:

--> **pol=poly([6 -5 1], ‘s’, ‘c’)**

pol =

$6 - 5s + s^2$

Definizione mediante le radici:

--> **pol=poly([2 3] ,’s’,’r’)** **oppure:** --> **pol=poly([2 3] ,’s’)**

pol =

$6 - 5s + s^2$

Definizione della variabile s:

--> **s=poly(0, ’s’);**

Le Radici o Zeri dei Polinomi (1)

⌘ Esempio: assegnato il **trinomio s^2-5s+6**

Definito mediante i coefficienti con lo statement:

--> **pol=poly([6 -5 1], 's', 'c')**

pol =

$6 - 5s + s^2$

il calcolo degli zeri è definito con l'istruzione :

--> **radici=roots(pol)**

radici =

2.

3.

Le Radici o Zeri dei Polinomi (2)

⌘ L'**argomento** della funzione **roots** può essere, a sua volta, ancora un'altra **funzione**:

Esempio: assegnato, mediante le sue **radici** o **zeri**, il polinomio $x^3-2x^2-29x+30$, gli **zeri** si rideterminano mediante l'istruzione:

```
-->radici1=roots(poly([6 -5  1],'x'))

 radici1  =

    1.
  - 5.
    6.
```

Le Radici o Zeri dei Polinomi (3)

- L'**argomento** della funzione **roots** può essere, a sua volta, ancora un'altra **funzione**:

  Esempio: assegnato, mediante i suoi **coefficienti,** il polinomio x^2-5x+6, gli **zeri** si ottengono con l'istruzione seguente:

```
-->radici2=roots(poly([6 -5  1],'x','c'))

    radici2  =

        2.

        3.
```

OPERAZIONI FRA POLINOMI (1)

- Si possono eseguire sui **polinomi** tutte le operazioni dell'algebra ordinaria: **somma**, **differenza**, **prodotto** e **quoziente**.

Esempio 1: assegnati, mediante i loro **coefficienti**, i polinomi (x^2-5x+6) e ($2x - x^2$), si ha:

```
-->pA=poly([6 -5 1],'x','c');
-->pB=poly([0 2 -1],'x','c');
-->polsom=pA+pB
   polsom=
          6 – 3x
```

OPERAZIONI FRA POLINOMI (2)

Esempio 2: assegnati, mediante i loro **coefficienti**, i **polinomi** (x^2-5x+6) e ($2x - x^2$), si ha:

```
-->pA=poly([6 -5 1],'x','c');
-->pB=poly([0 2 -1],'x','c');
-->pdiffer1=pA – pB
   pdiffer1 =
```

$$6 - 7x + 2x^2$$

```
-->pdiffer2=pB – pA
   pdiffer2=
```

$$-6 + 7x - 2x^2$$

OPERAZIONI FRA POLINOMI (3)

Esempio 3: assegnati, mediante i loro **coefficienti**, i **polinomi** ($x^2 - 5x+6$) e ($2x - x^2$), si ha:

```
-->pA=poly([6 -5 1],'x','c');
```
$\Rightarrow (6 - 5x + x^2)$

```
-->pB=poly([0 2 -1],'x','c');
```
$\Rightarrow (2x - x^2)$

```
-->polprod=pA*pB
    polprod=
```
$$12x - 16x^2 + 7x^3 - x^4$$

```
-->polpotenza=pA^2
```
(equivale a pA*pA)

```
    polpotenza=
```
$$36 - 60x + 37x^2 - 10x^3 + x^4$$

OPERAZIONI FRA POLINOMI (4)

Esempio 4: assegnati, mediante i loro **coefficienti**, i **polinomi** (x^3+3x^2-x-3) e $(x-3)$, si ha:

```
-->pA=poly([-3 -1 3 1],'x','c');
```
$\Rightarrow (-3-x+3x^2+x^3)$

```
-->pB=poly([3  1],'x','c');
```
$\Rightarrow (3+x)$

```
-->poldiv=pdiv(pA,pB)
   poldiv=
```
$-1+x^2 \quad \Rightarrow$ (polinomio quoziente)

```
-->[polR,polQ]=pdiv(pA,pB)
```
polQ= $-1+x^2 \quad \Rightarrow$ (polinomio quoziente)

polR= $0 \quad \Rightarrow$ (polinomio resto)

OPERAZIONI SUI POLINOMI (5)

Esempio 5: assegnato, mediante i suoi **coefficienti**, il **polinomio** **(x^3+3x^2-x-3):**

```
-->pA=poly([-3 -1 3 1],'x','c');
```

$\Rightarrow$ ($-3-x+3x^2+x^3$)

con le diverse istruzioni che seguono si ottiene:

```
-->V=coeff(pA)
    V= -3  -1  3  1
```

(sono dati i coefficienti del polinomio)

```
-->grado=degree(pA)
    grado= 3
```

(viene restituito il grado del polinomio)

```
-->der=derivat(pA)
```

der=$-1+6x+3x^2$ (viene fatta la derivata del polinomio)

OPERAZIONI SUI POLINOMI (6)

Esempio 6: assegnato, mediante i suoi **coefficienti**, il **polinomio** **($x^4 + 10x^3 + 35x^2 + 50x + 24$):**

```
-->pA=poly([24  50  35  10  1],'x','c');
```

con le diverse istruzioni che seguono si ottiene:

```
-->sv=varn(pA)
```

sv= x **(restituisce il simbolo della variabile)**

```
-->[fattori]=factors(pA)
```

fattori(1) ➔ 1+x **(sono dati i fattori del polinomio)**

fattori(2) ➔ 2+x **pA=(1+x)·(2+x)·(3+x)·(4+x)**

fattori(3) ➔ 3+x

fattori(4) ➔ 4+x

POLINOMIO CARATTERISTICO

- **Assegnata la matrice quadrata $A_{(n,n)}$, l'istruzione poly(A,'x') fornisce il "polinomio caratteristico" della matrice nella variabile specificata, ovvero, se la variabile è "x", si ha: p(x)=det(A-xI).**

Esempio: assegnata la **matrice A**

--> A=[0 1; –2 –3];

l'istruzione seguente:

--> polcarat=poly(A,'x')

fornisce il **polinomio caratteristico**:

polcarat=

$$2 + 3x + x^2 \rightarrow \boxed{\det(A - xI)}$$

POLINOMIO CARATTERISTICO

⌘ **Assegnata la "matrice quadrata" $A_{(n,n)}$, allora il polinomio caratteristico p(s)=det(A-sI) ad essa associato, nella indeterminata specificata s, può determinarsi, anche, con le seguenti istruzioni:**

Esempio: assegnata la **matrice A**

--> A=[3 1; 2 2];

le istruzioni seguenti:

--> s=poly(0,'s');

--> AL=A-s*eye();

--> polcarat=det(AL)

$$AL = \begin{pmatrix} 3-s & 1 \\ 2 & 2-s \end{pmatrix}$$

forniscono il **polinomio caratteristico**:

polcarat=

$$4 - 5s + s^2$$

$$\det(AL) = \det(A - sI)$$

POLINOMIO CARATTERISTICO

- Sia assegnato, mediante i suoi **coefficienti** o le sue **radici**, un **polinomio monico** interpretabile come il **polinomio caratteristico** di una **matrice quadrata** di **ordine n** uguale al **grado** del **polinomio** stesso.

  L'istruzione **companion** consente di determinare una matrice **quadrata A** di **ordine n** che **ammette** come **polinomio caratteristico** il polinomio assegnato.

  Le istruzioni seguenti:

  -->pol=poly([4 -5 1],'s','c'); $\quad (s^2-5s+4)$

  -->A=companion(pol)

  forniscono la **matrice quadrata** di **ordine 2, A(2,2)** definita da:

$$\boxed{A=\begin{bmatrix}5 & -4\\ 1 & 0\end{bmatrix}}$$

POLINOMIO CARATTERISTICO

In generale, assegnato il polinomio caratteristico monico $P(\lambda)$ di grado n:

$$P(\lambda) = a_0+a_1 \lambda +a_2 \lambda^2 +a_3 \lambda^3 +\ldots+a_{n-1} \lambda^{(n-1)} + \lambda^n$$

l'istruzione:

--> A=companion(P(λ))

fornisce la matrice quadrata $A_{(n,n)}$ – di ordine n – avente gli elementi a_{ij} che corrispondono alla seguente struttura:

blocco diagonale

$$A=\begin{pmatrix} -a_{n-1} & -a_{n-2} & \ldots & -a_1 & -a_0 \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

POLINOMIO CARATTERISTICO

Esempio: assegnati i due polinomi caratteristici:

P1= $s^3 - 6s^2 + 11s - 6$

P2= $s^4 + 4s^3 + 3s^2 + 2s + 1$

le istruzione seguenti:

--> A1=companion(P1)   --> A2=companion(P2)

forniscono, rispettivamente, le matrici:

$$A_2 = \begin{pmatrix} -4 & -3 & -2 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad A_1 = \begin{pmatrix} 6 & -11 & 6 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

blocchi diagonali

Le Rappresentazioni grafiche mediante i grafici 2D (1)

- In **scala lineare** → ***plot*** oppure ***plot2d***
  plot(x,y) traccia il grafico dei punti che hanno sulle ascisse gli elementi del **vettore x** e sulle ordinate gli elementi del **vettore y**
- In **scala semilogaritmica** o **logaritmica** → ***plot2d*** mediante l'uso della ***logflag='cc'***.
- Diagrammi **polari di Nyquist** e **cartesiani di Bode** → ***nyquist***, ***bode***.

Rappresentazione grafica con GRAFICI 2D (bidimensionali) (2)

- In **scala lineare** → ***plot (x,y,'mc')***
  traccia il **grafico** dei **punti** che hanno come **ascisse** gli elementi del **vettore x** nonché come **ordinate** gli elementi del **vettore y**, utilizzando come tratto il **mark** individuato dal **carattere 'm'** ed il **colore** definito dal **carattere 'c'**.

Codice associato al carattere 'c'

y=yellow	**m=magenta**	**c=cyan**	**r=red**
g=green	**w=white**	**b=blue**	**k=black**

Rappresentazione grafica con GRAFICI 2D (bidimensionali) (3)

- Codice associato al carattere ‘m’

.	point	x	x-mark	+	plus
*	star	s	square	d	diamond
v	triangle (down)			^	triangle (up)
<	triangle (left)			>	triangle (right)
0	circle			p	pentagram
-	solid	:	dotted		
-.	dashdot	--	dashed		

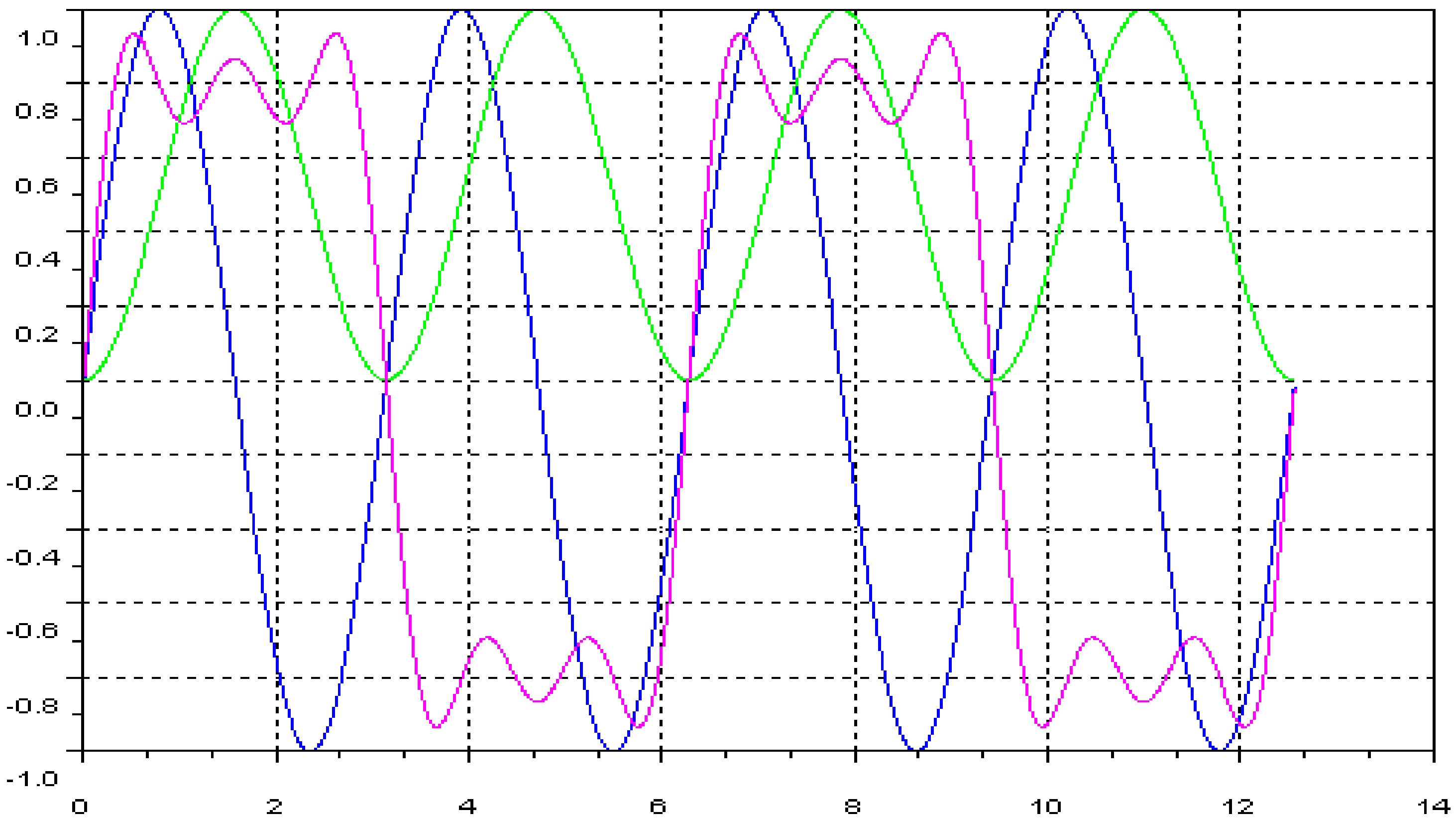
Rappresentazione grafica con GRAFICI 2D (bidimensionali) (4)

Esempio: il seguente codice

```
-->t=0:0.01:4*%pi;
-->y1=sin(2*t);
-->y2=sin(t).^2;
-->y3=sin(t)+(1/3)*sin(3*t)+(1/5)*sin(5*t);
-->plot(t,y1,'b',t,y2,'g',t,y3,'m'),xgrid
```

evidenzia nella finestra di Scilab Graphic, che risulta attualmente attiva, il grafico seguente:

Rappresentazione grafica con GRAFICI 2D (bidimensionali) (4 bis)



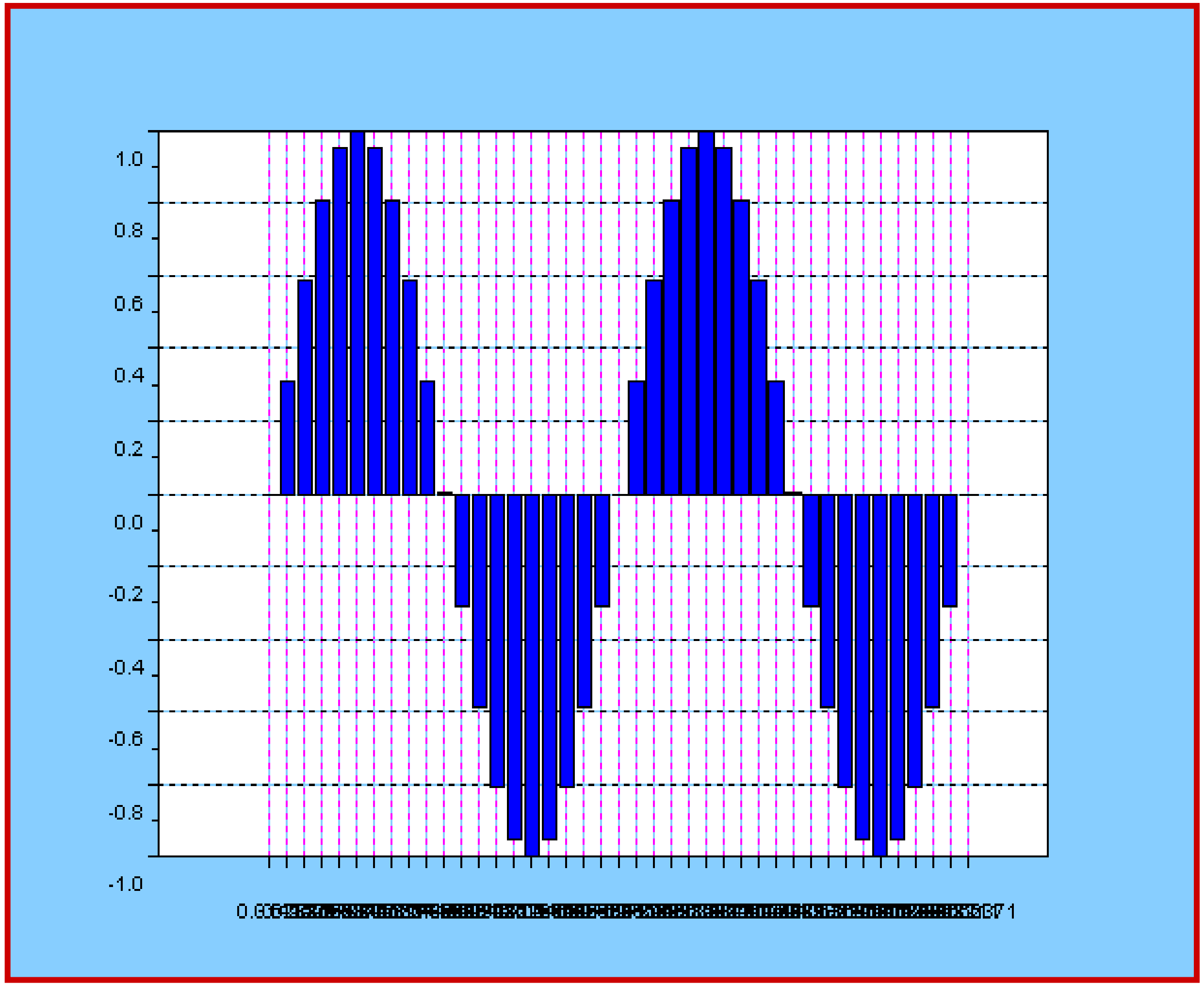
Rappresentazione dei grafici
GRAFICI A BARRE (1)

⌘ È anche possibile ottenere il tracciamento dei **grafici a barre** tramite il comando **bar** **mediante** la seguente sintassi:

Esempio 1: **il codice riportato:**

```
-->t=0:0.1*%pi:4*%pi;
-->y=sin(t);
-->bar(t,y),xgrid
```

Rappresentazione dei grafici
GRAFICI A BARRE (2)



Fornisce nella finestra grafica di Scilab, che è al momento attiva, il grafico a Barre che viene di lato mostrato

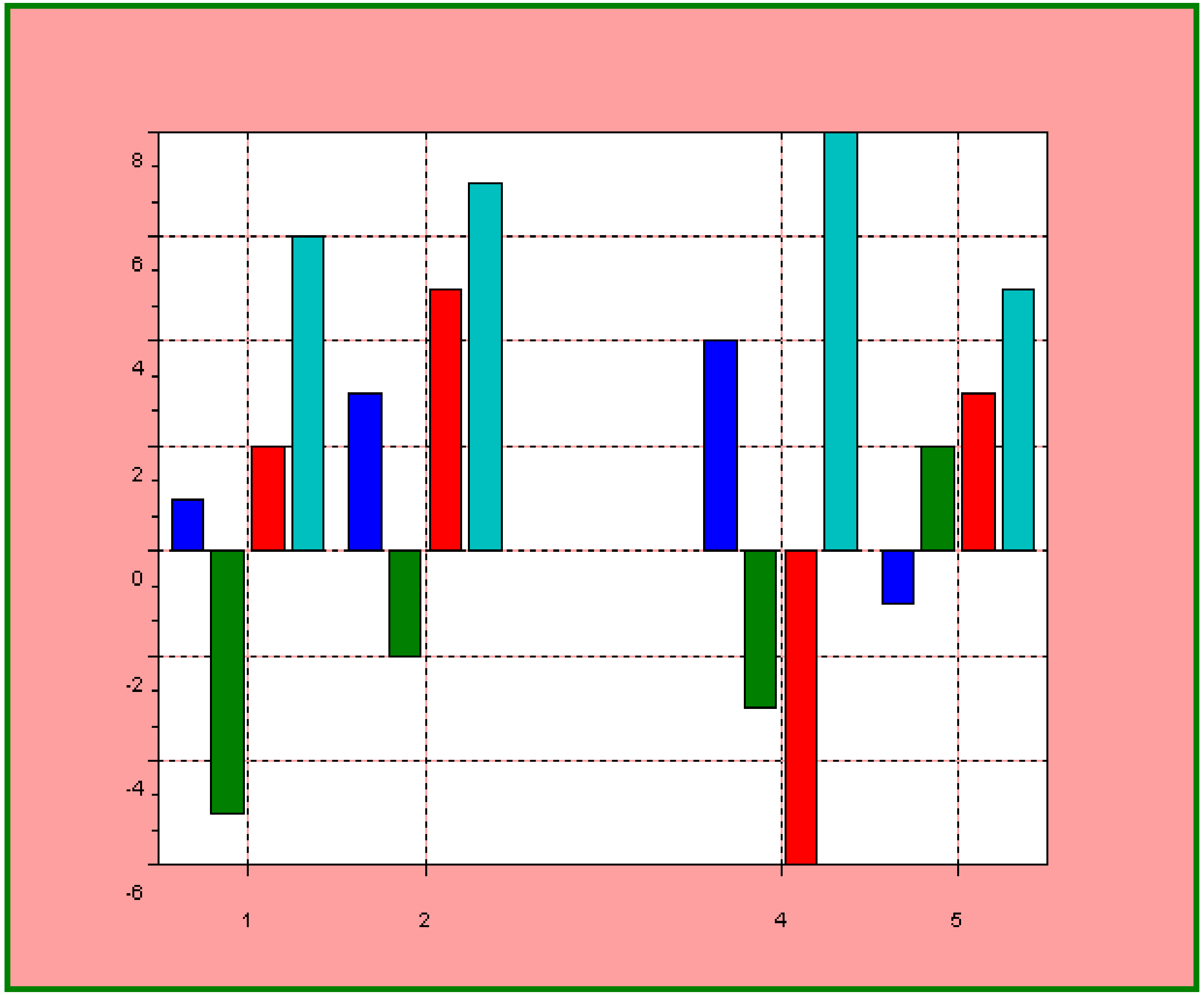
Rappresentazione dei grafici
GRAFICI A BARRE (3)

- Si può tracciare anche *la rappresentazione grafica di una distribuzione in classi di un carattere continuo*, cioè un **ISTOGRAMMA**

Esempio 2: il codice di seguito riportato:

```
--> z=[1 2 4 5];
--> y=[1 -5 2 6;3 -2 5 7;4 -3 -6 8;-1 2 3 5];
--> bar(z,y), xgrid
```

Rappresentazione dei grafici
GRAFICI A BARRE (4)



fornisce nella finestra grafica di Scilab, che è al momento attiva, l'**istogramma** (*barre colorate*) che viene a lato mostrato

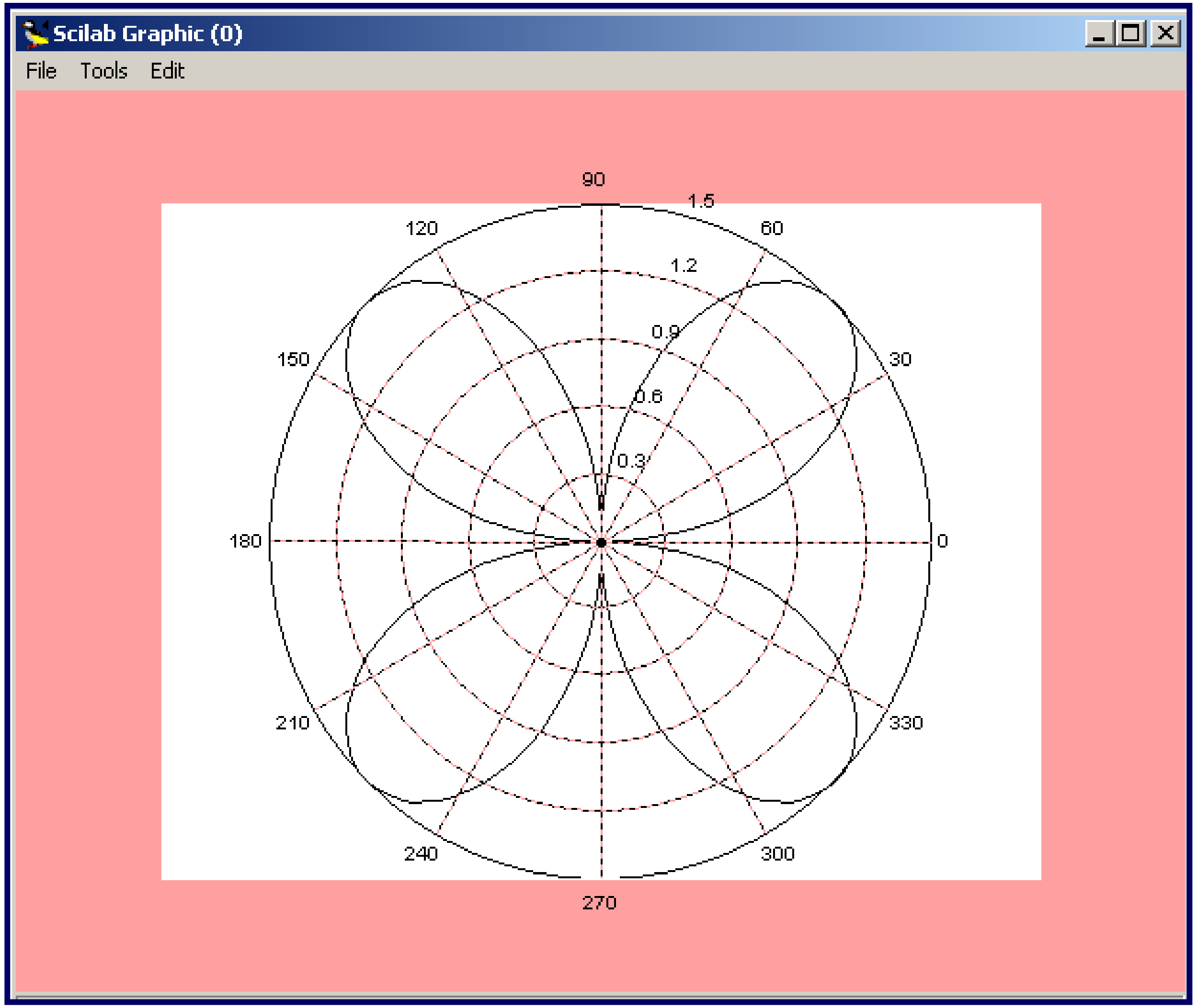
Rappresentazione dei grafici 2D in COORDINATE POLARI (1)

- È anche possibile tracciare i grafici in coordinate polari, **diagrammi polari**, con il comando **polarplot**, tramite la seguente sintassi:

Esempio: il seguente codice

```
-->theta=0:0.01*%pi:2*%pi;
-->rho=3*sin(theta).*cos(theta);
-->polarplot(theta,rho)
```

Rappresentazione dei grafici 2D in COORDINATE POLARI (2)



fornisce nella finestra grafica di Scilab, che è al momento attiva, il grafico polare che viene mostrato qui a lato

Rappresentazione GRAFICI 2D
Istruzione "plot2dn" (1)

L'istruzione più recente e avanzata per i grafici bidimensionali presenta la seguente sintassi:

plot2dn(x, y, <option>)

n = 1: grafico con linea a tratto continuo (obsoleto)

n = 2: grafico a tratti di linea costanti (spezzata – piecewise);

n = 3: grafico a barre verticali;

n = 4: grafico con tratti di linea espressi in stile vettore "→";

- **Se n viene omesso (missing), il grafico è tracciato, per default, in scala lineare con linea continua;**
- **se compare l'opzione logflag='cc', il grafico viene tracciato con scale logaritmiche (logarithmic plotting) in dipendenza dei due caratteri che costituiscono la stringa posta fra gli apici.**

Rappresentazione GRAFICI 2D
Istruzione "plot2dn" (2)

In sostanza l'opzione logflag='cc' agisce come di seguito esplicitato:

logflag='nl': il grafico viene tracciato con le ascisse in scala lineare e le ordinate in scala logaritmica;

logflag='ln': il grafico è visualizzato con le ascisse in scala logaritmica e le ordinate in scala lineare;

logflag='ll': il grafico viene tracciato con ascisse e ordinate entrambe in scala logaritmica;

logflag='nn': il grafico viene visualizzato con le ascisse e le ordinate in scala lineare; tale opzione per il grafico lineare risulta, pertanto, ridondante;

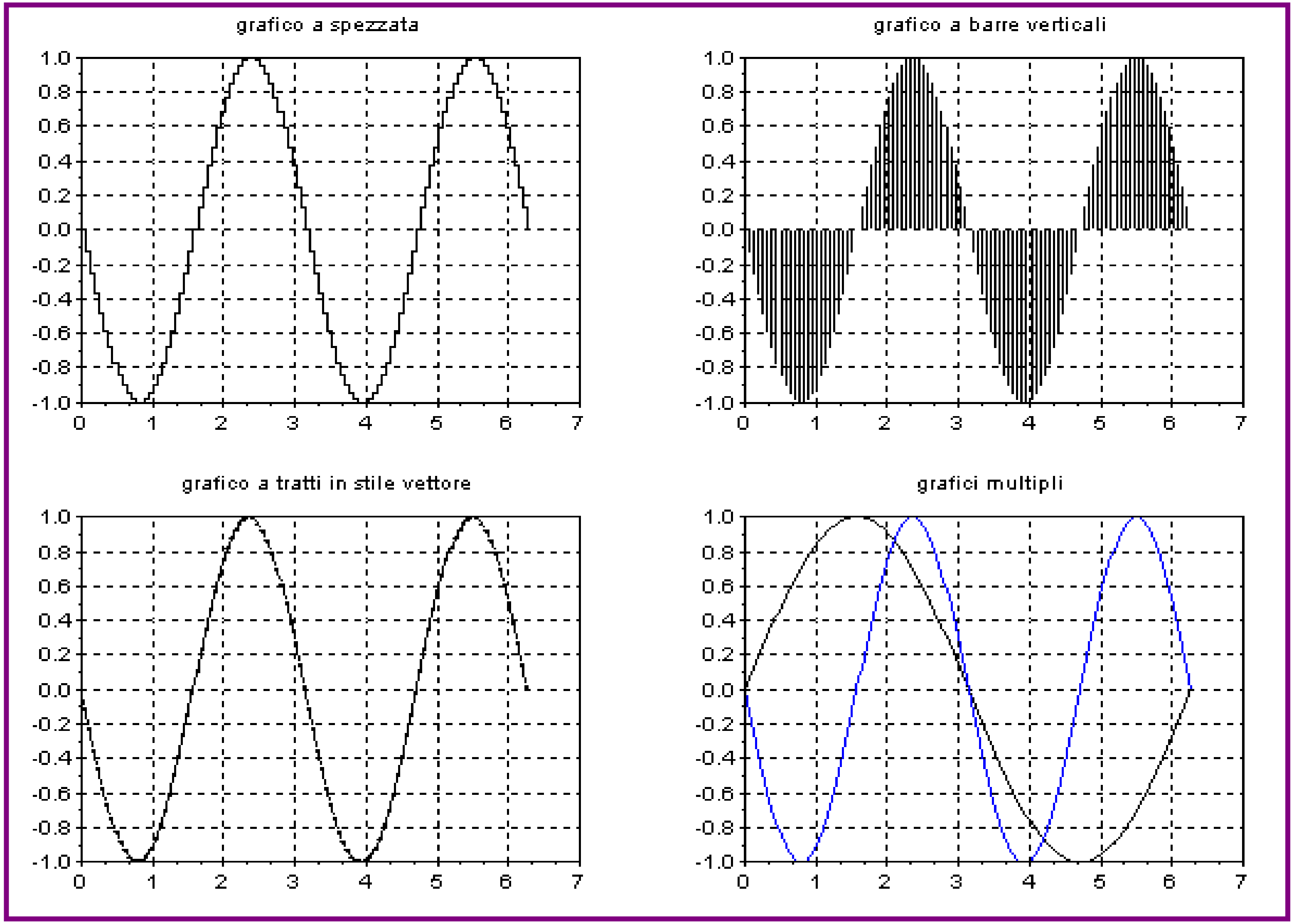
Rappresentazione GRAFICI 2D
Istruzione "plot2dn" (3)

- Come esempio applicativo dell'istruzione, si propone il seguente listato in codice Scilab:

```
-->t=0:0.02*%pi:2*%pi;
-->y1=sin(2*t+%pi);
-->a=‘grafico a spezzata’;b=‘grafico a barre verticali’;
-->c=‘grafico a tratti in stile vettore’;
-->d=‘grafici multipli’;
-->subplot(221),plot2d2(t,y1),xgrid;xtitle(a);
-->subplot(222),plot2d3(t,y1),xgrid;xtitle(b);
-->subplot(223),plot2d4(t,y1),xgrid;xtitle(c);
-->x=0:0.01*%pi:2*%pi;
-->y2=sin(x); y3=sin(2*x+%pi);
-->subplot(224),plot2d1(x,[y2’ y3’]),xgrid;xtitle(d);
```

GRAFICI 2D: istruzione plot2dn (4)

I grafici che si ottengono sono i seguenti:



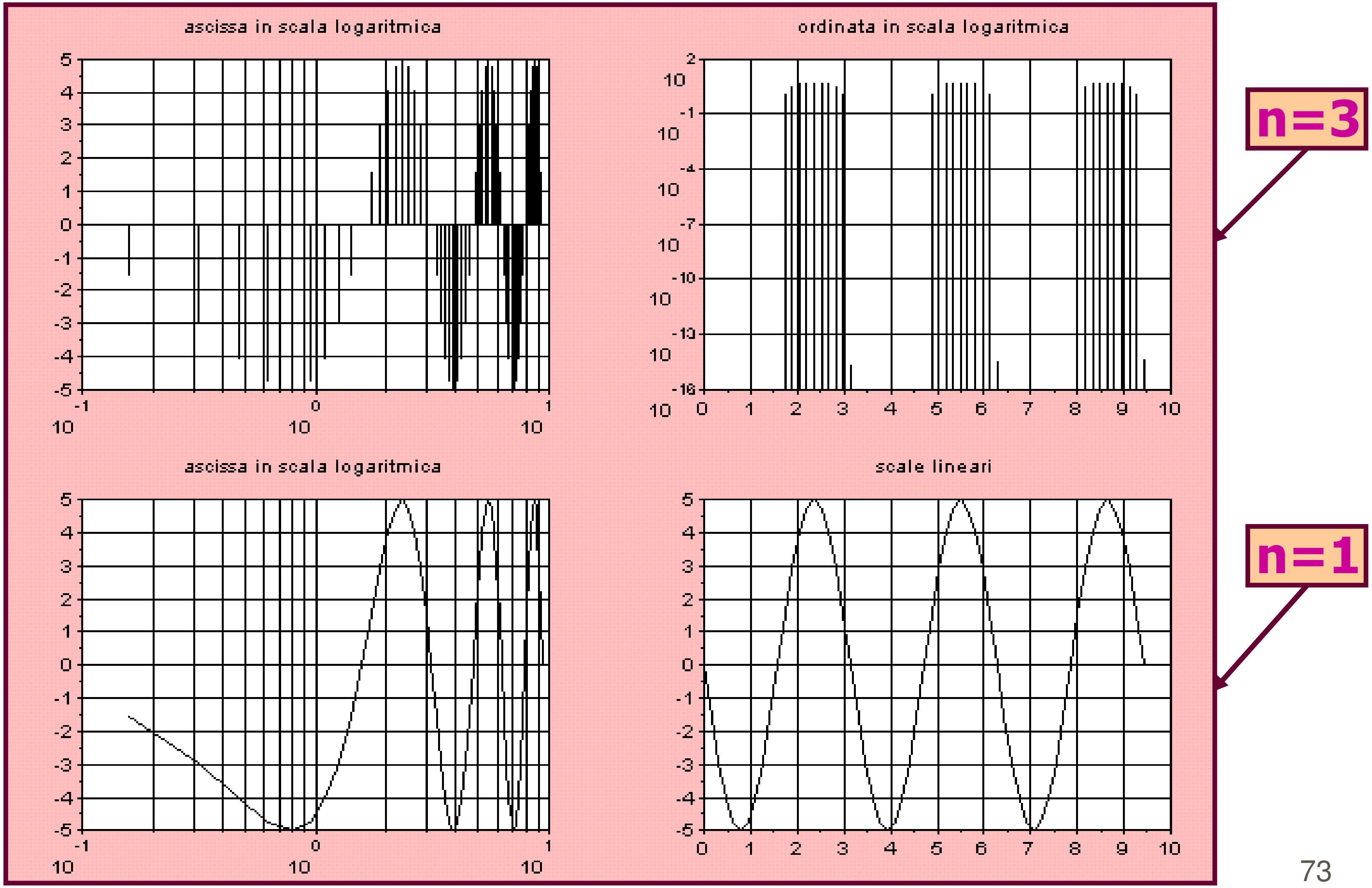
Rappresentazione GRAFICI 2D
Istruzione "plot2dn" e logflag (5)

- Come secondo esempio applicativo, si vuole proporre il seguente listato in codice Scilab:

```
-->x=0:0.05*%pi:3*%pi;
-->y=5*sin(2*x+%pi);
-->a='ascissa in scala logaritmica';
-->b='ordinata in scala logaritmica';
-->d='scale lineari';
-->subplot(221),plot2d3(x,y,logflag='ln'),xgrid;xtitle(a);
-->subplot(222),plot2d3(x,y,logflag='nl'),xgrid;xtitle(b);
-->subplot(223),plot2d1(t,y,logflag='ln'),xgrid;xtitle(a);
-->subplot(224),plot2d1(t,y,logflag='nn'),xgrid;xtitle(d);
```

GRAFICI 2D: istruzione plot2dn (6)

I grafici che si ottengono sono i seguenti:



Rappresentazione GRAFICI 2D
grafici multipli con plot2dn (7)

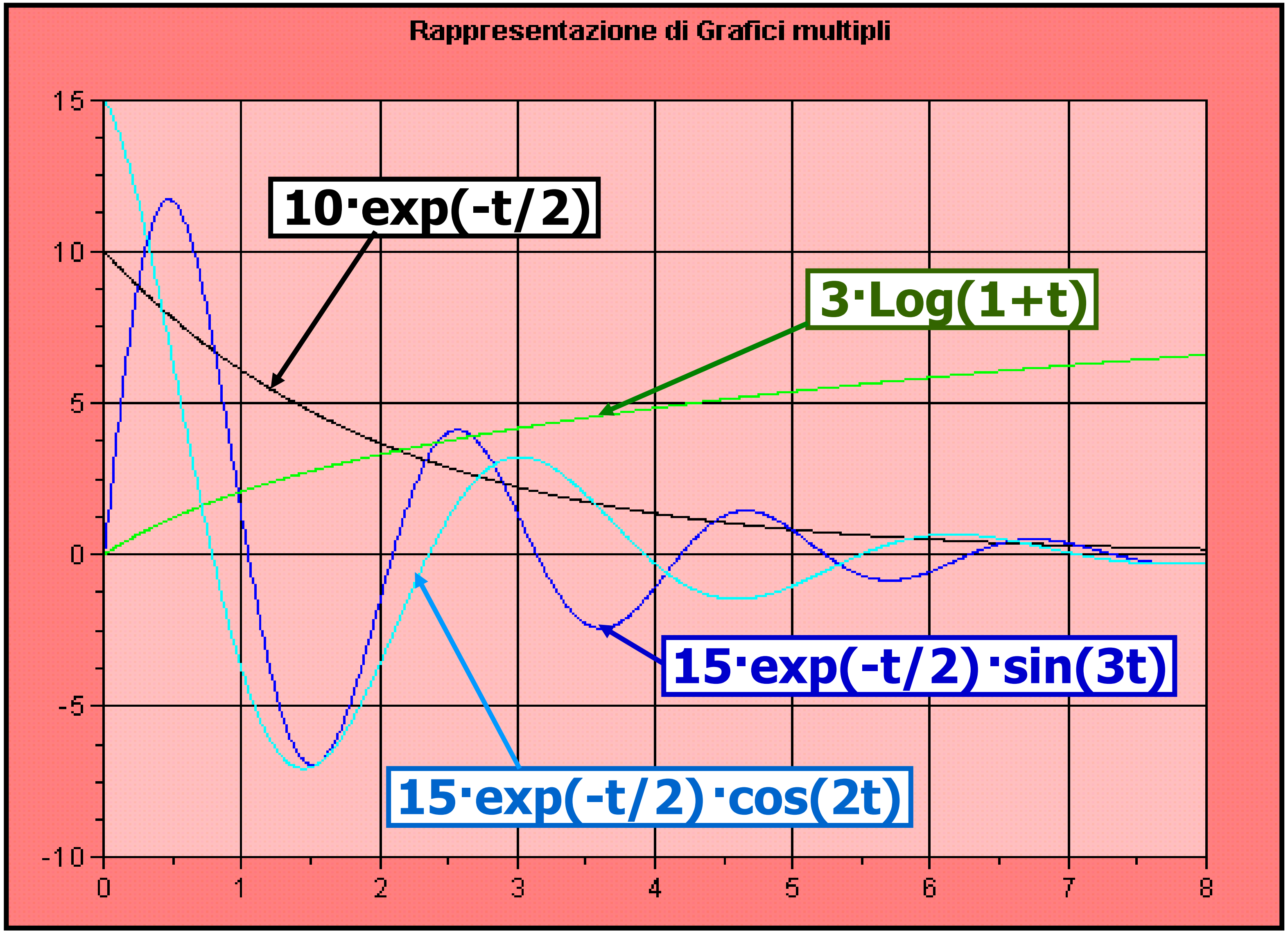
Come terzo esempio applicativo, si desidera esaminare il caso in cui si hanno più funzioni da rappresentare sullo stesso grafico.

All'uopo, si osservi il seguente listato SCILAB:

```
-->t=0:0.01:8;
-->y1=10*exp(-t/2);
-->y2=15*exp(-t/2).*sin(3*t);
-->y3=3*log(1+t);
-->y4=15*exp(-t/2).*cos(2*t);
-->a=‘Rappresentazione di Grafici multipli’;
-->plot2d(t,[y1',y2',y3',y4']),xgrid;title(a)
```

Grafici multipli con plot2dn (8)

Esso produce il seguente grafico multiplo



Sistemi Dinamici Lineari

- Un **Sistema Dinamico Lineare Tempo Invariante** può essere descritto:
  - nella forma di **variabili di stato** mediante **quattro matrici A**, **B**, **C**, **D**;
  - nella forma di **funzione di trasferimento**, mediante i **due polinomi N(s)** e **D(s)**.
- In **Scilab**, perciò, è possibile definire i **sistemi lineari** come **oggetti** a partire da entrambe le descrizioni.

Definizione dei sistemi lineari
(a tempo continuo)

- Mediante lo **spazio degli stati**:
  - definire le **matrici A**, **B**, **C**, **D**;
  - definire il **sistema** con il **comando** ***syslin***.

- Mediante la **funzione di trasferimento**:
  - definire i **polinomi N(s)**=**num** e **D(s)**=**den** (**numeratore** e **denominatore** della **f.d.t.**);
  - definire il **sistema** con il **comando** ***syslin***.

Esempio 1: Spazio degli Stati

- Definizione del **sistema** nello **Spazio degli Stati**

$$\dot{x} = -x + 3u$$

$$y = 4x + 2u$$

```
-->A=-1; B=3; C=4; D=2;
-->sistema=syslin('c',A,B,C,D)
 sistema  =
      sistema(1)  (state-space
   system:)
!lss  A  B  C  D  X0  dt  !
       sistema(2) = A matrix =
  - 1.
       sistema(3) = B matrix =
   3.
       sistema(4) = C matrix =
   4.
       sistema(5) = D matrix =
   2.
       sistema(6) = X0 (initial state) =
   0.
       sistema(7) = Time domain =
 c
```

Esempio 2: Funzione trasferimento

⌘ Definizione del sistema lineare: **ordue**

```
-->num=poly([1 1],'s','c');
-->den=poly([16 3 1],'s','c');
-->ordue=syslin('c',num,den);
 ordue  =

        1 + s
   ---------------
   16 + 3s + s²
```

$$G(s)=\frac{s+1}{s^2+3s+16}$$

Conversione fra i due modelli

- Passaggio dal modello **Spazio degli Stati** al modello **Funzione di Trasferimento** si ottiene mediante il comando che attiva la:

  funzione ss2tf

- Passaggio dal **modello Funzione di Trasferimento** al modello **Spazio degli Stati** avviene mediante il comando che attiva la:

  funzione tf2ss

Conversione: Spazio degli Stati → Funzione di Trasferimento (1)

- Passaggio dal modello **Spazio degli Stati** al modello **Funzione di Trasferimento**:

```
-->A=[-3 -16;1 0];B=[1;0];C=[1 1]; D=0;
-->x0=[1;0];
-->secordss=syslin('c',A,B,C,D,x0);
-->secordtf=ss2tf(secordss)
```

$$\text{secordtf} = \frac{1+s}{16+3s+s^2}$$

Conversione: Spazio degli Stati → Funzione di Trasferimento (2)

⌘ Passaggio dal modello **Spazio degli Stati** al modello **Funzione di Trasferimento**:

```
-->A=[-3 -16;1 0];B=[1;0];C=[1 1]; D=0;
-->x0=[1;0];
-->secordss=syslin('c',A,B,C,D,x0);
-->[d,num,den]=ss2tf(secordss)
```

si ottiene: **den** = $16+3s+s^2$

num = $1+s$

d = 0

Conversione:
Funzione di Trasferimento → Spazio degli Stati (1)

⌘Passaggio dal **modello Funzione di Trasferimento** al modello **Spazio Stati**:

```
--> num=poly([1 1],'s','c');
--> den=poly([16 3 1],'s','c');
--> pippotf=syslin('c',num,den);
--> pipposs =tf2ss(pippotf)
```

si ottiene:

Conversione:
Funzione di Trasferimento → Spazio degli Stati (2)

pipposs(1) (state-space system:)

|lss A B C D X0 dt |

pipposs(2) = A matrix = - 2.1386139 - 1.3861386

10.213861 - 0.8613861

pipposs(3) = B matrix = - 1.4071951

0.1407195

pipposs(4) = C matrix = - 0.7106335 - 2.082D-16

pipposs(5) = D matrix = 0.

pipposs(6) = X0 (initial state) = 0. 0.

pipposs(7) = Time domain = c

Sistemi Dinamici Lineari

La determinazione dei **poli** e degli **zeri** di un **sistema lineare** espresso mediante la funzione di trasferimento **G(s)**, avviene sempre col comando **roots** tramite la seguente sintassi:

-->poli=roots(giesse.den)

-->zeri=roots(giesse.num)

in cui **giesse** esprime la **funzione** di **trasferimento** del sistema lineare in esame, precedentemente definita nei diversi modi ormai noti. Se G(s) è stata definita nello **spazio degli stati**, deve essere prima trasformata, con la **funzione ss2tf** nella **rappresentazione** della **FdT**.

Sistemi Dinamici Lineari

Esempio 3: Determinare poli e zeri del sistema lineare definito nello spazio degli stati dalle matrici **A**,**B**,**C** e **D**.

```
-->A=[-3 -16; 1 0];B=[1;0];C=[1 1];D=0;
-->GSS=syslin('c',A,B,C,D);
-->GTF=ss2tf(GSS)
 GTF =

          1 + s
    ----------------
      16 + 3s + s²
```

$$G(s) = \frac{1+s}{16+3s+s^2}$$

```
-->poliGTF=roots(GTF.den)
```

$p_{1,2}$=-1.5±3.7081i

```
-->zeriGTF=roots(GTF.num)
```

z_1=-1

Simulazione di sistemi lineari

- Funzioni disponibili per la simulazione:
  - ***imp*** -> simulazione **risposta all'impulso**;
  - ***step*** -> simulazione **risposta a scalino**;
  - ***csim*** -> simulazione con **ingresso** qualsiasi e **stato iniziale qualsiasi**.
- Sintassi della funzione **csim**:

```
-->y=csim('step',t,sistema);
-->y=csim('imp',t,sistema);
-->[y,x]=csim('imp',t,sistema);
```

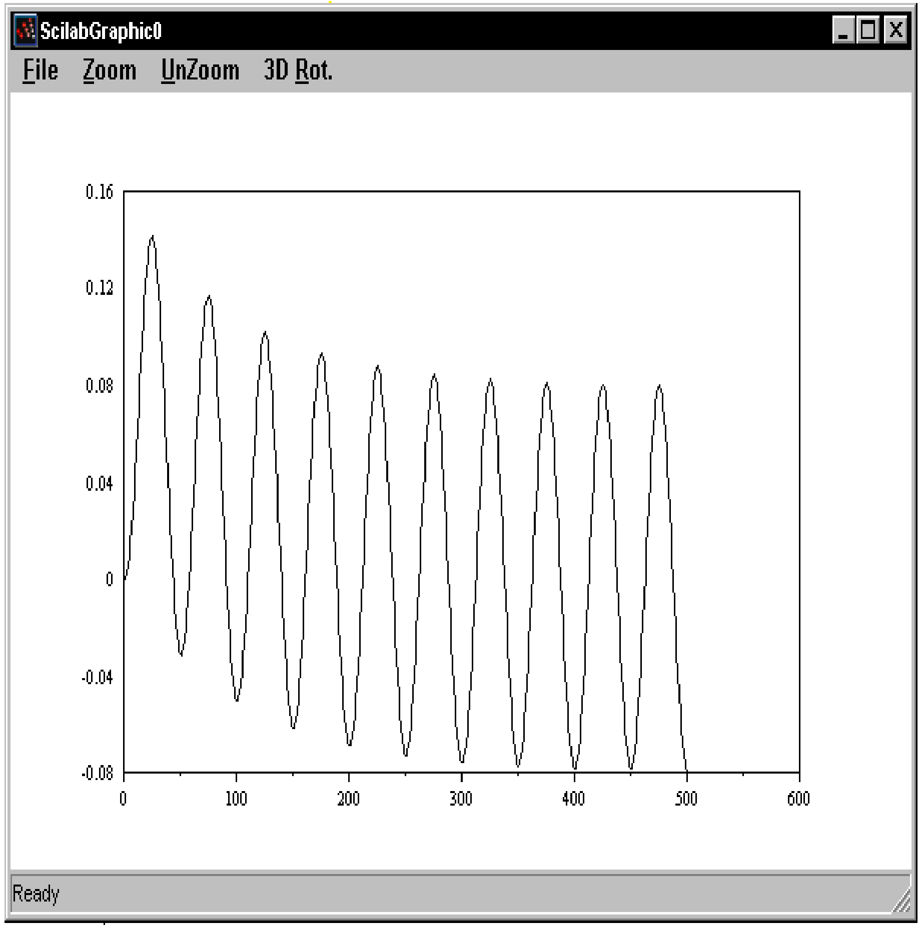
Vettore dello stato

Vettore dei tempi

Esempio
Risposta al segnale sinusoidale

$$u(t) = sen(2\pi f t)$$

```
-->s=poly(0, 's');
-->prova=syslin('c',1/(1+s));
-->t=(0:0.01:5);
-->u=sin(2*%pi*2*t);
-->y=csim(u, t, prova);
-->plot(t, y)
```



Istruzioni per i Diagrammi cartesiani di BODE

- **bode(sistema)**
- **bode(sistema,fmin,fmax)**
- **bode(sistema,frq)**
- **bode(frq,db,phi)**
- **bode([sistema1;sistema2])**
- **bode([sistema1;sistema2],fmin,fmax)**
- **bode([sistema1;sistema2],frq)**

Se **fmin** e **fmax** sono omesse, scilab assume per default **fmin=0.001 Hz** ed **fmax=1000 Hz**

Rappresentazione grafica Diagrammi di BODE

- Si consideri il **sistema lineare** del **2° ordine** e di **tipo 1** (un **polo** è sito nell'**origine)**, avente funzione di trasferimento:

$$G(s) = \frac{10}{s \cdot (1+s)}$$

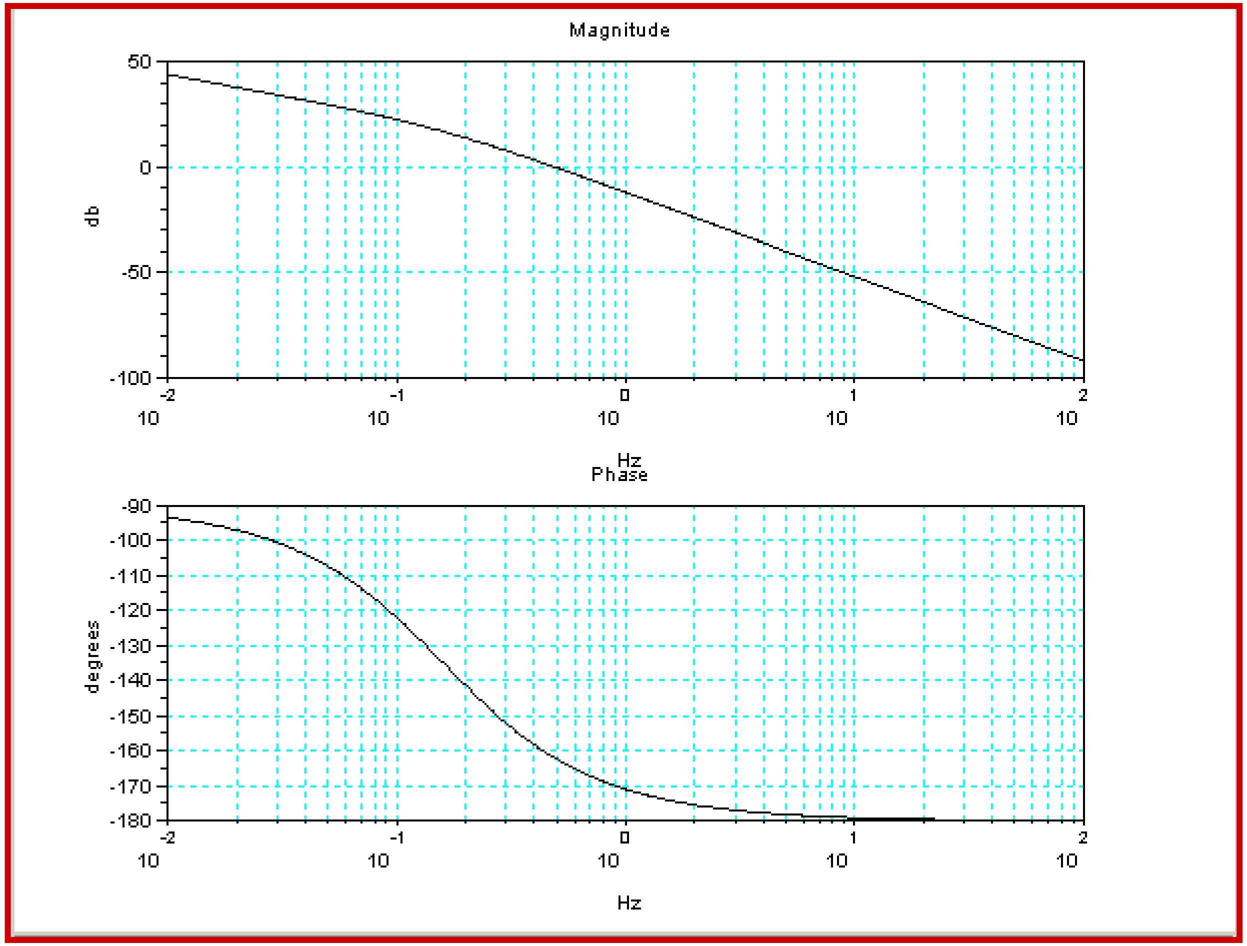
Per ottenere il **diagramma** di **BODE** del **modulo** e della **fase** di **G(jω)** si devono dare le seguenti **istruzioni**, nella finestra di lavoro di **Scilab**:

```
num=10;
den=poly([0 –1],'s','r');
giesse1=syslin('c',num,den);
W=logspace(-2,2,500);
Bode(giesse1,w)
```

Diagrammi di BODE

⌘ Si ottengono i due **grafici** di seguito riportati, noti come **diagrammi cartesiani** di **BODE** del **modulo** e della **fase** di **G(jω)**:

$$G(s) = \frac{10}{s \cdot (1+s)}$$



Esempio: Diagrammi di BODE

Con le istruzioni, in codice Scilab, contenute nel programma che segue .....

```
num1=poly([1 10],'s','c');
den1=poly([-0.5 -1],'s','r');
giesse1=syslin('c',num1,den1);
num2=1;
den2=poly([4 1 1],'s','c');
giesse2=syslin('c',num2,den2);
bode([giesse1;giesse2])
```

$$G_1(s)=\frac{1+10s}{(1+2s)\cdot(1+s)}$$

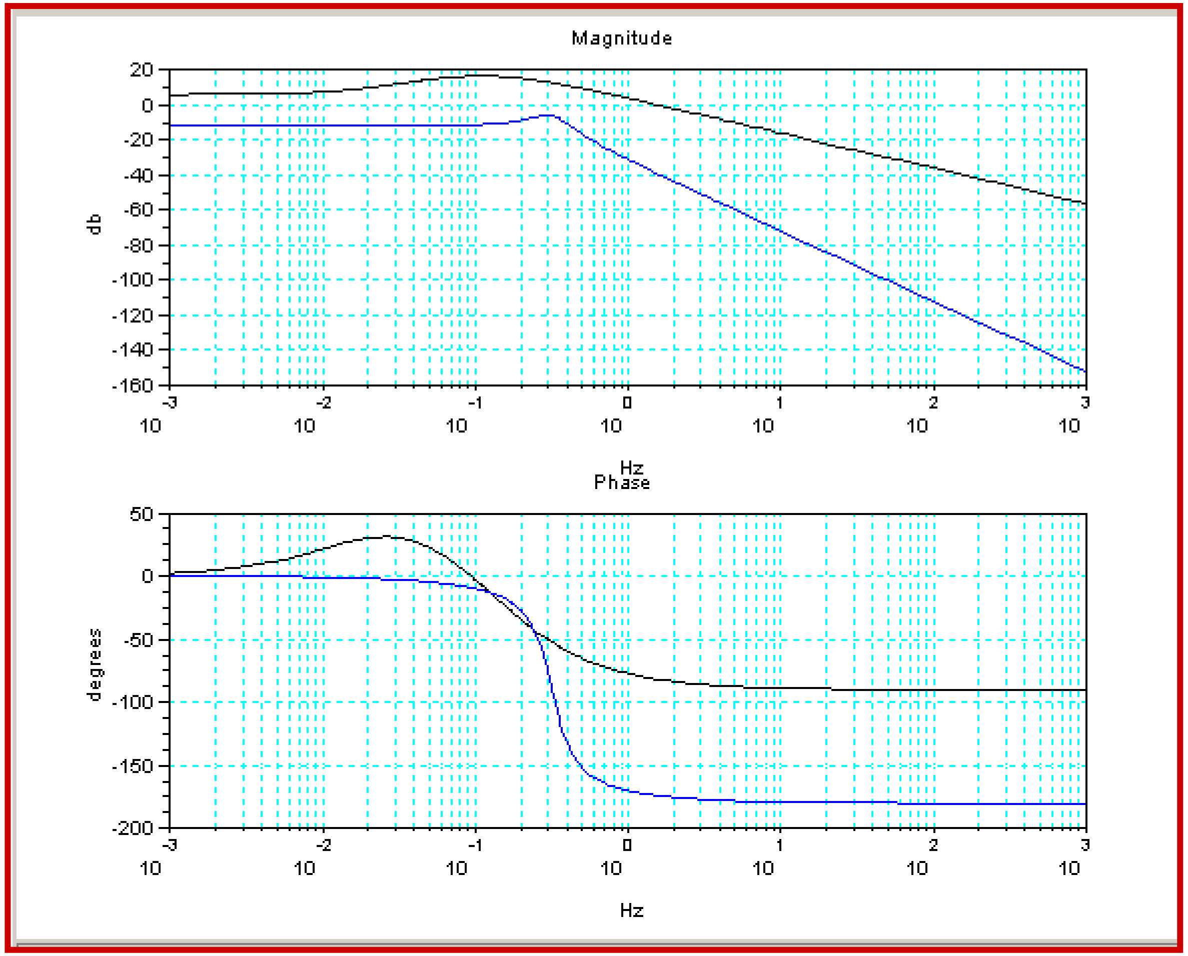
$$G_2(s)=\frac{1}{4+s+s^2}$$

Diagrammi di BODE

- Si ottengono i **diagrammi cartesiani** di **BODE** del **modulo** e della **fase** di **$G_1(j\omega)$** e di **$G_2(j\omega)$:**

$$G_1(s) = \frac{1+10s}{(1+2s)\cdot(1+s)}$$

$$G_2(s) = \frac{1}{4+s+s^2}$$



Istruzioni per i Diagrammi Polari e di NYQUIST

- **nyquist(sistema)**
- **nyquist(sistema,fmin,fmax)**
- **nyquist(sistema,frq)**
- **nyquist(frq,db,phi)**
- **nyquist([sistema1;sistema2])**
- **nyquist([sistema1;sistema2],fmin,fmax)**
- **nyquist([sistema1;sistema2],frq)**

Se **fmin** e **fmax** sono omesse, scilab assume per default **fmin=0.001 Hz** ed **fmax=1000 Hz**

Rappresentazione diagramma Polare e di NYQUIST

⌘ Si consideri il **sistema lineare** del "**3° ordine**" e di "**tipo 0**", (nessun **polo** è sito nell'**origine)**, avente **funzione di trasferimento**:

$$G(s) = \frac{10}{(1+s)^3}$$

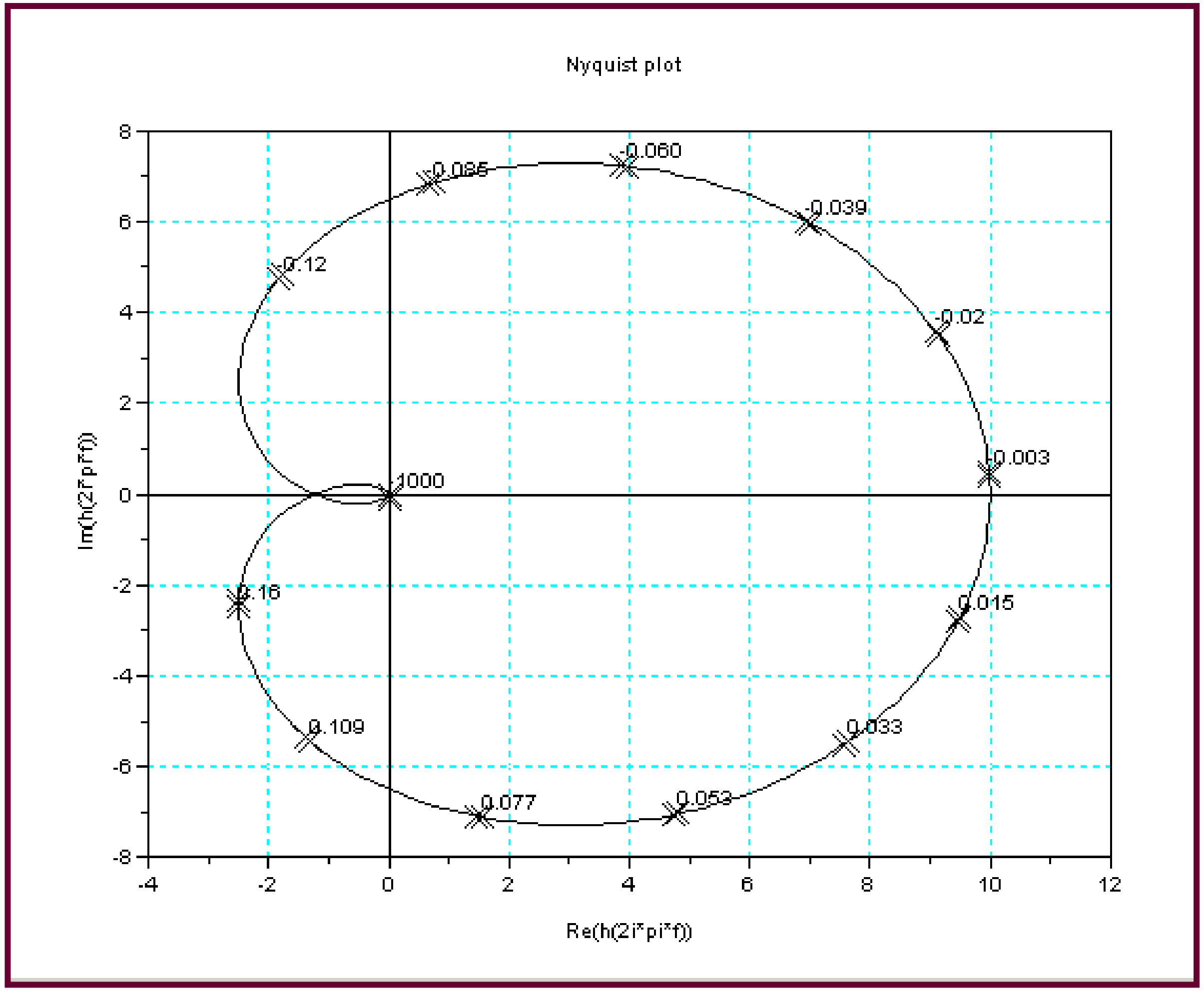
Per ottenere il **diagramma Polare e di Nyquist** necessita assegnare, nella finestra di lavoro, le seguenti **istruzioni,** di codice **Scilab**:

```
num=10;
den=poly([-1 -1 -1],'s','r');
giesse=syslin('c',num,den);
nyquist(giesse)
```

Diagramma di NYQUIST

- Si ottiene il **Diagramma Polare** ed il **Diagramma di Nyquist** della funzione di **risposta armonica G (jω)**:

$$G(s) = \frac{10}{(1+s)^3}$$



Margine di Fase
Margine di Guadagno (1)

- I parametri caratteristici del grado di stabilità di un Sistema in reazione negativa sono forniti dalle seguenti istruzioni:

[gm,fr]=g_margin(sistema)

in cui è: **gm = guadagno margine espresso in decibel**

fr = frequenza in Hertz alla quale il diagramma di Nyquist attraversa l'asse reale negativo, nota anche come f_π.

La pulsazione corrispondente alla fase Φ_π= -180° è definita dall'istruzione che realizza l'operazione di calcolo seguente:

ω_π=2*fr*%pi

Margine di Fase
Margine di Guadagno (2)

- I parametri caratteristici del grado di stabilità di un Sistema in reazione negativa sono forniti dalle seguenti istruzioni:

[phm,fr]=f_margin(sistema)

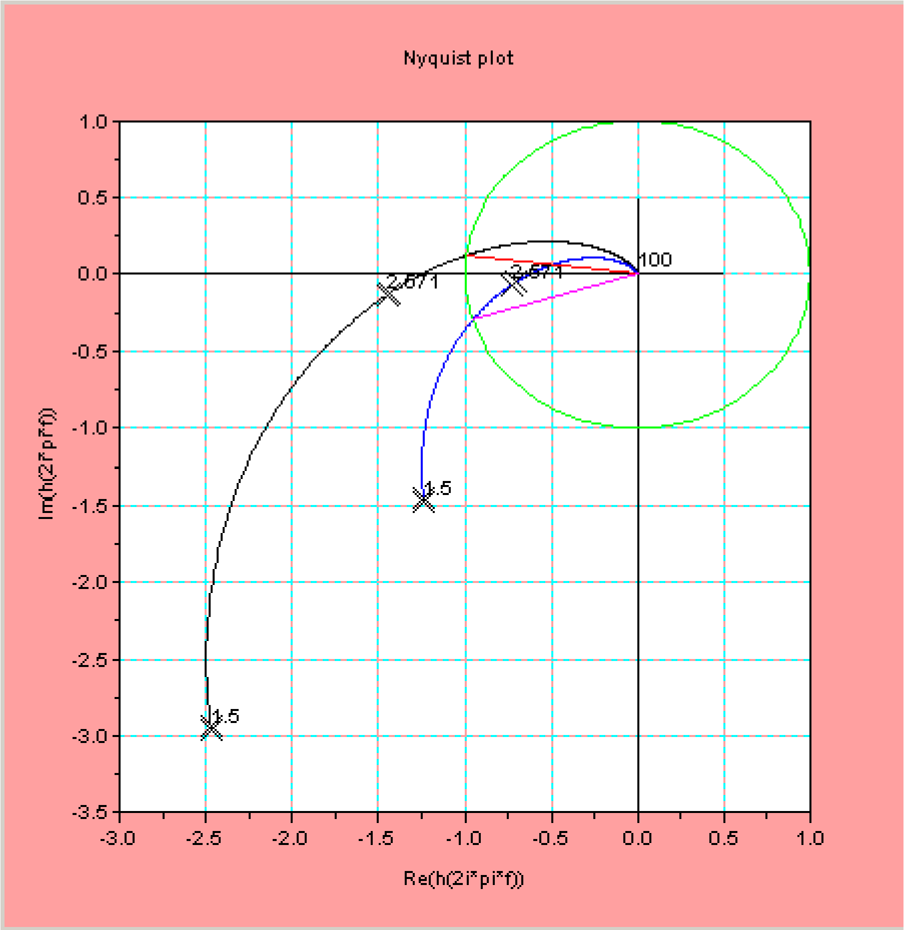
in cui è: **phm = fase crtica Φ_C; fase calcolata alla pulsazione critica ω_C, pulsazione alla quale il diagramma di Bode dei moduli taglia l'asse a 0 dB**

fr = frequenza, in Hertz, alla quale il diagramma di Nyquist interseca il cerchio di raggio unitario.

La pulsazione ω_C corrispondente alla fase critica Φ_C è definita dall'istruzione che realizza l'operazione di calcolo seguente:

ω_C=2*fr*%pi

Frequenza critica, Pulsazione critica e Fase critica



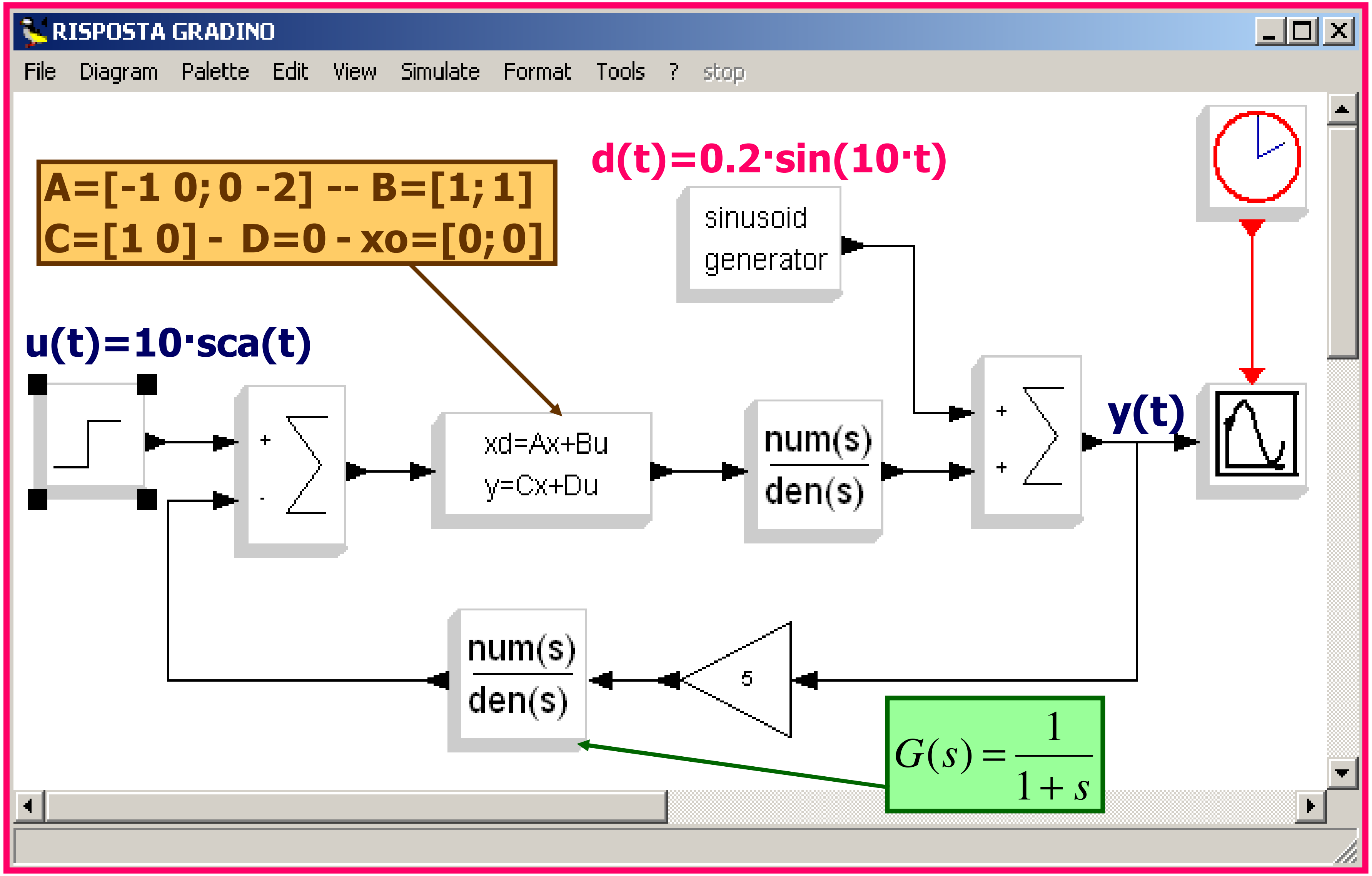
Φ_C= fase critica = è la fase calcolata alla pulsazione critica ω_C, pulsazione alla quale il diagramma di Bode dei moduli taglia l'asse a 0 dB (decibel)

fc = frequenza critica, data in Hertz, è la frequenza alla quale il diagramma di "Nyquist" interseca il cerchio di raggio unitario

La pulsazione ω_C afferisce alla fase critica Φ_C

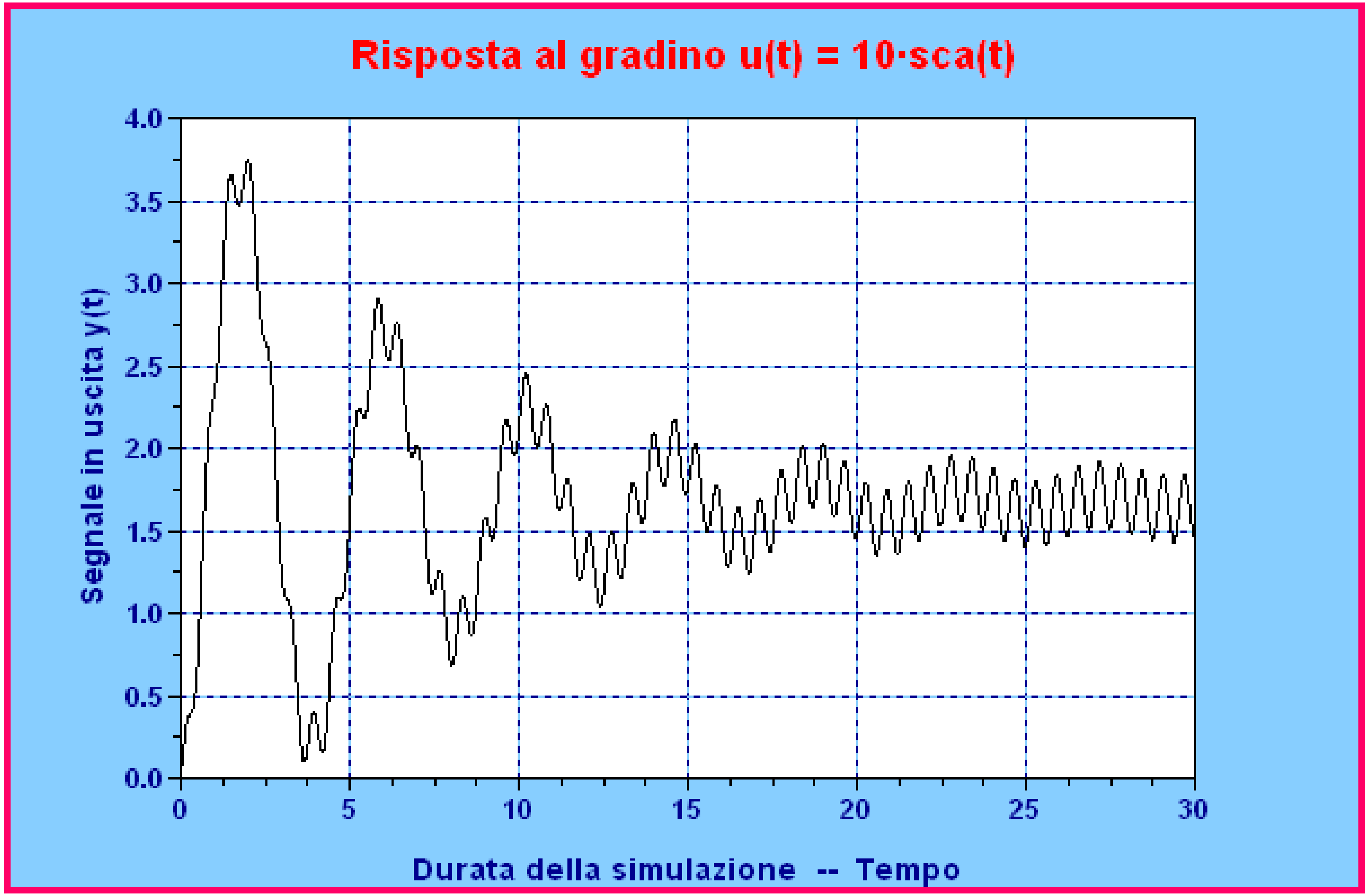
$$\omega_C = 2 \cdot \pi \cdot f_C$$

Introduzione a SCICOS



Introduzione a SCICOS

La simulazione in ambiente Scicos genera il grafico seguente



L'ambiente Scicos

- **Scicos** (**Scilab Connected Object Simulator**): è l'**ambiente grafico** per la simulazione di sistemi **complessi**.
- **Perché non basta Scilab**?
  - E' spesso necessario **simulare** dei **sistemi complessi**, composti da numerosi **blocchi interconnessi** tra loro.
  - Spesso i singoli blocchi sono **non-lineari** o **tempo-varianti**.
  - Può essere necessario dover integrare **blocchi continui** e **discreti**.

Principio di funzionamento (1)

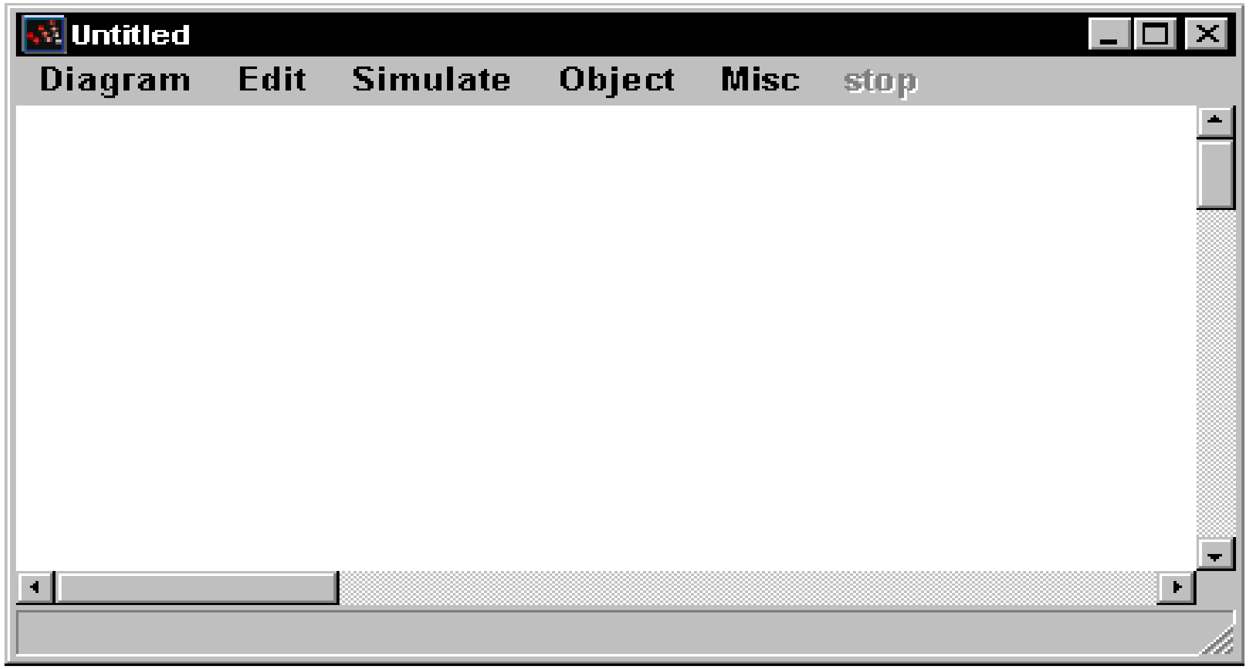
- **Scicos contiene una libreria di blocchi** che descrivono elementi **statici** e **dinamici elementari**
- **L'utente compone lo schema a blocchi del sistema da simulare tramite l'interconnessione dei blocchetti elementari**.
- **Scicos genera automaticamente le equazioni e risolve il problema numerico di simulazione desiderato**.

Principio di funzionamento (2)

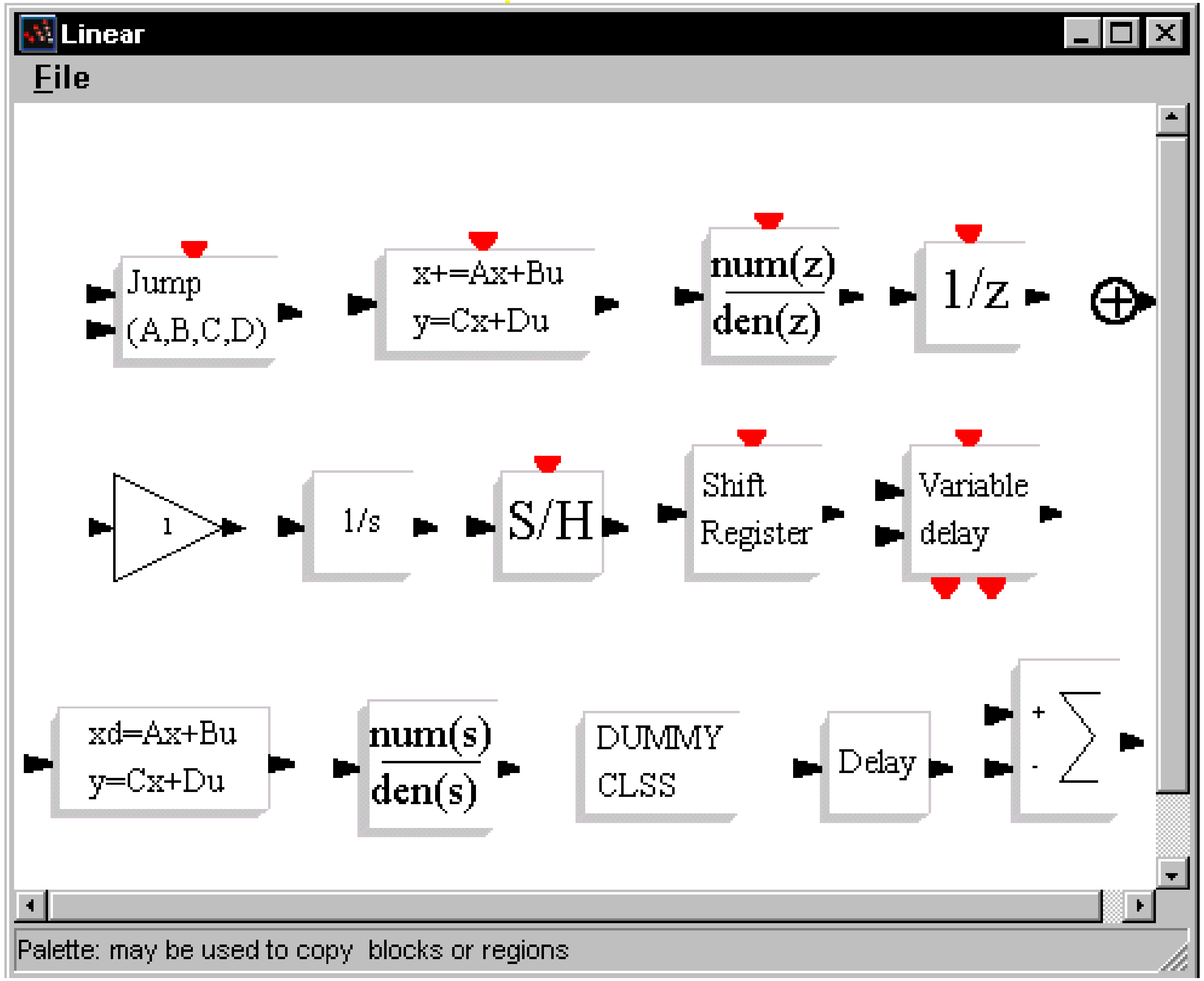
- **Scicos interagisce con Scilab attraverso la Scilab window;**
- **ciò significa che i modelli Scicos possono contenere variabili definite nella Scilab window.**

L'Interfaccia Grafica

- Digitando ***scicos();*** al prompt di **Scilab**, si apre un nuovo modello (foglio bianco) di **Scicos** ed è possibile **comporre il sistema da simulare** mediante i diversi **blocchi** disponibili.

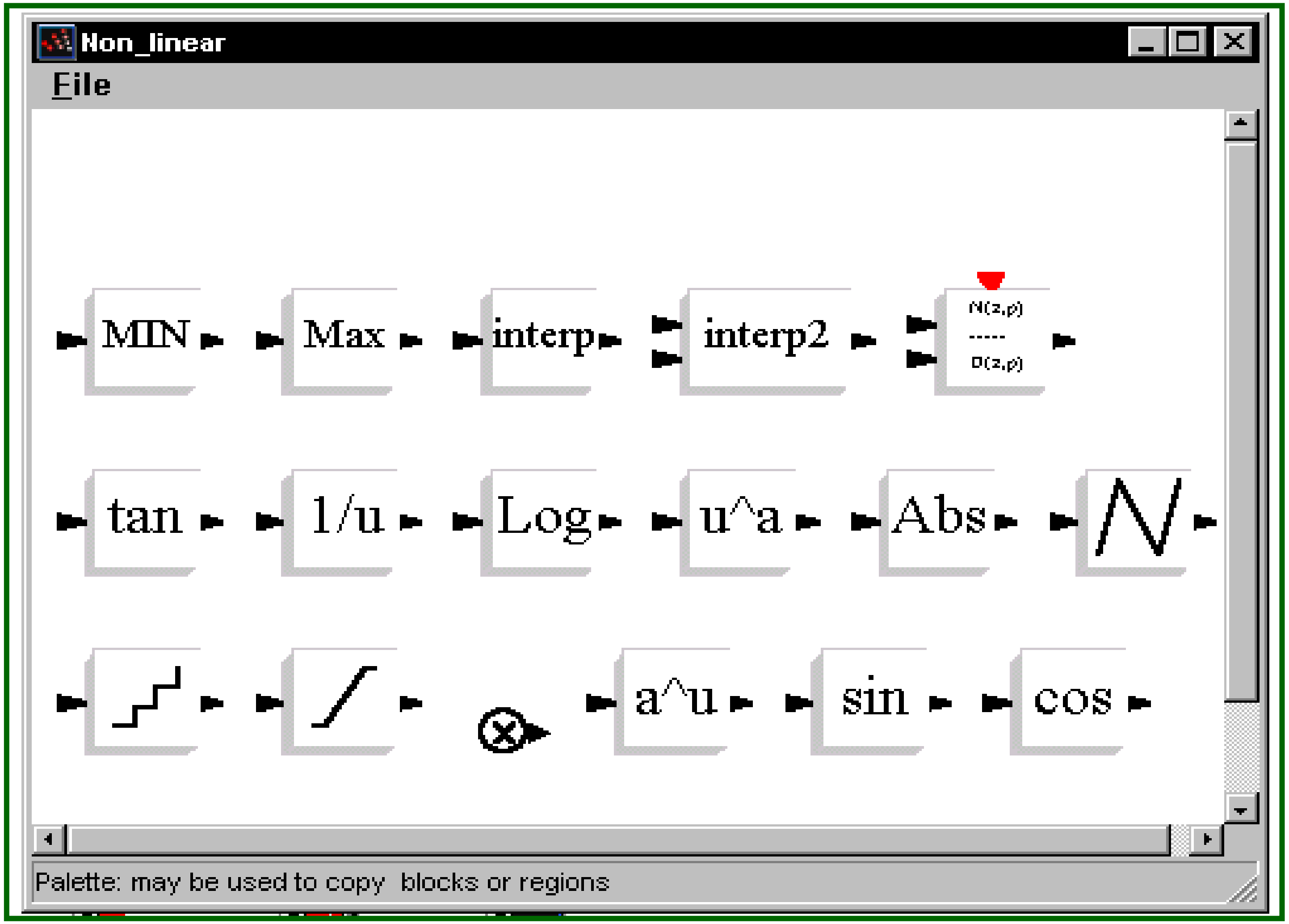


Principali librerie Scicos (1)



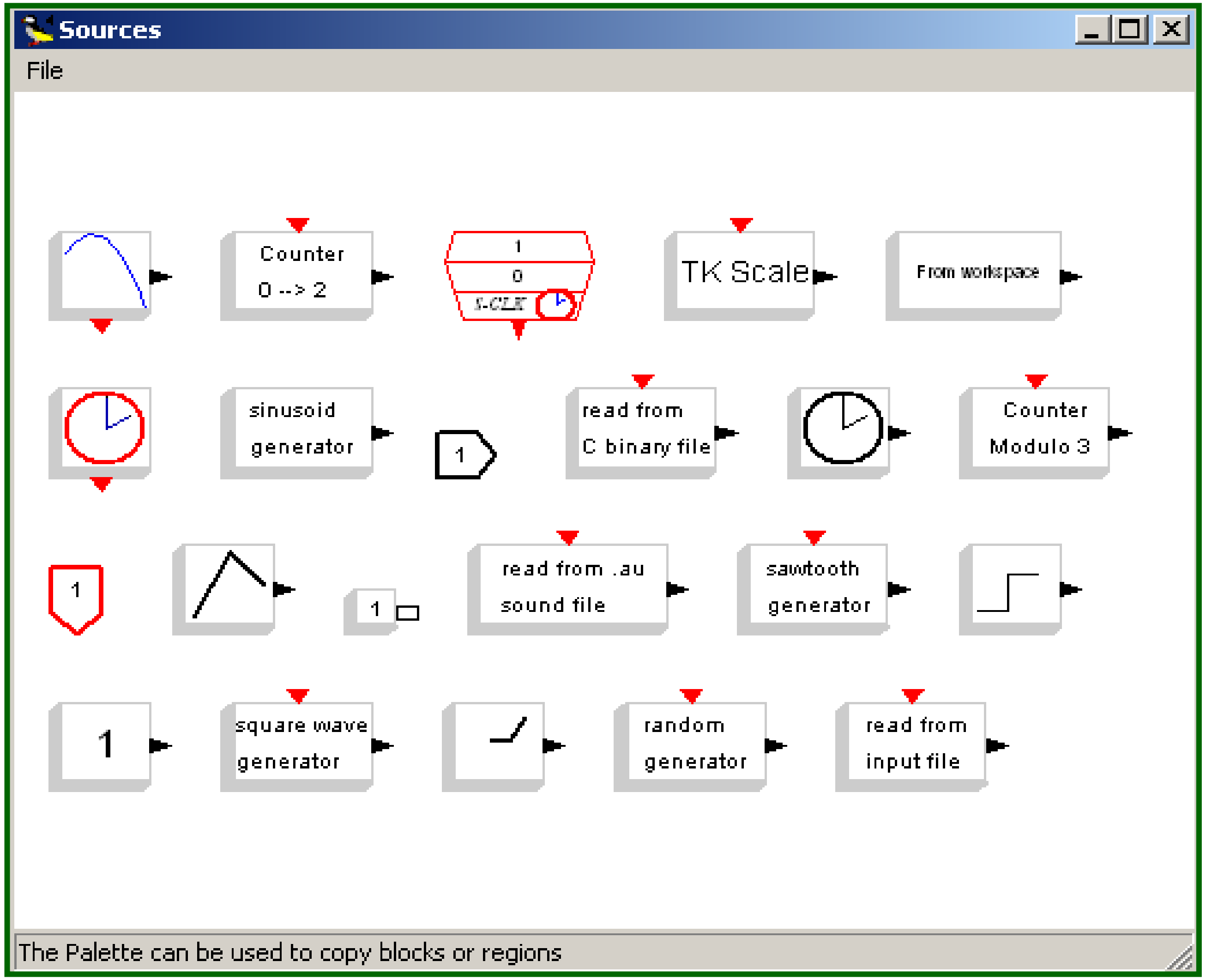
Blocchi lineari - (menù Linear)

Principali librerie Scicos (2)



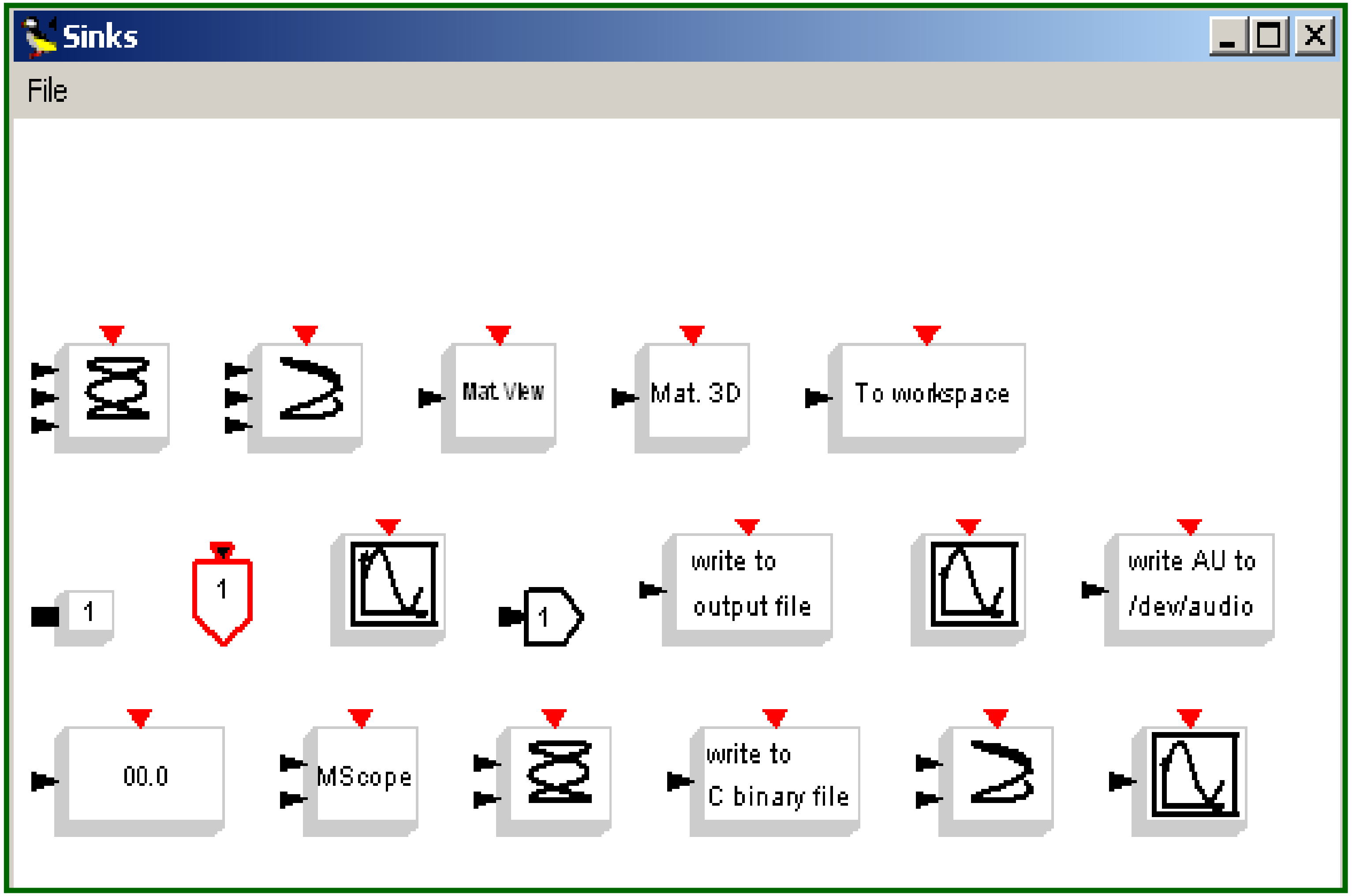
Blocchi non lineari -- (menù Non-linear)

Principali librerie Scicos (3)



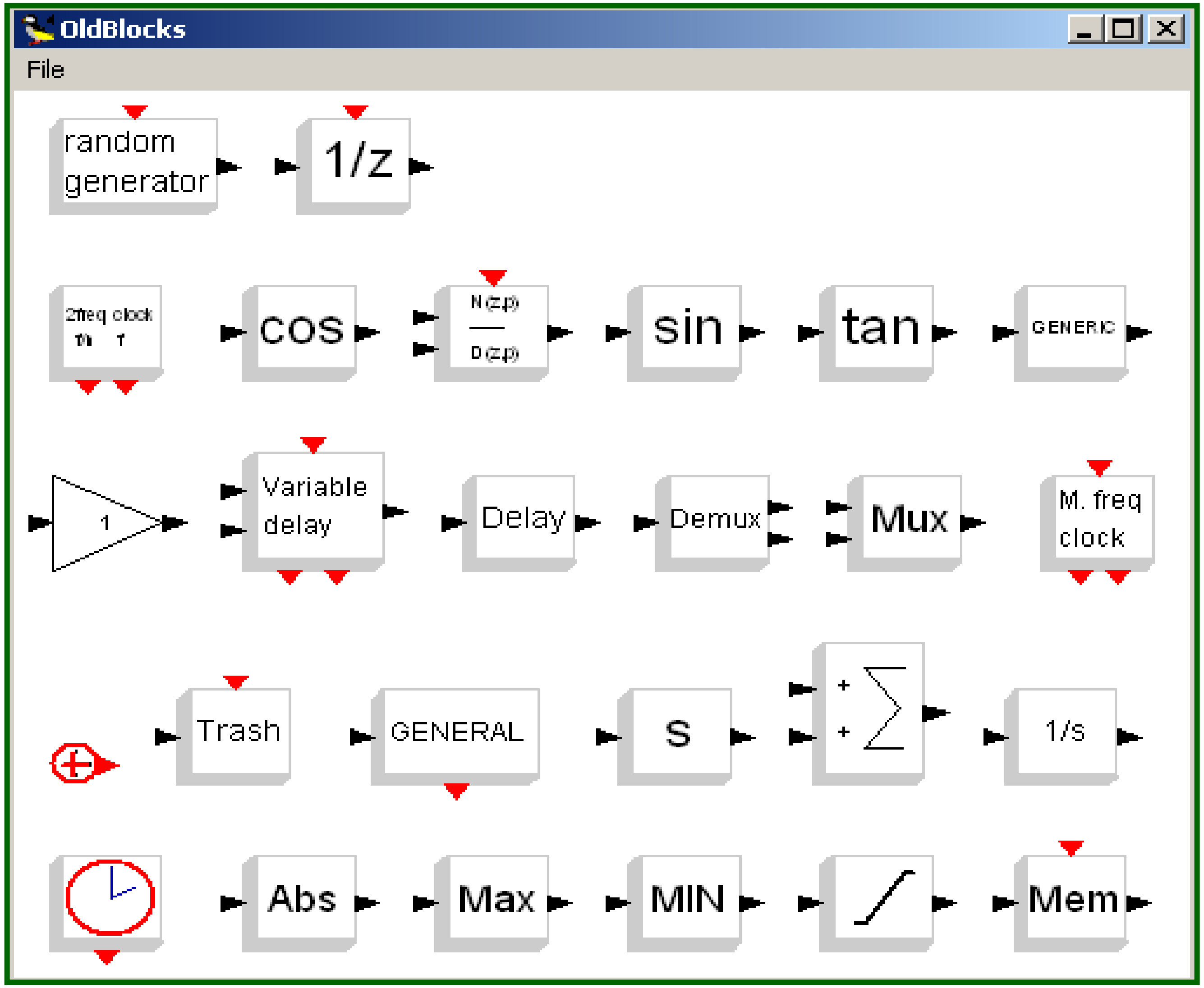
Segnali di ingresso - (menù Sources)

Principali librerie Scicos (4)



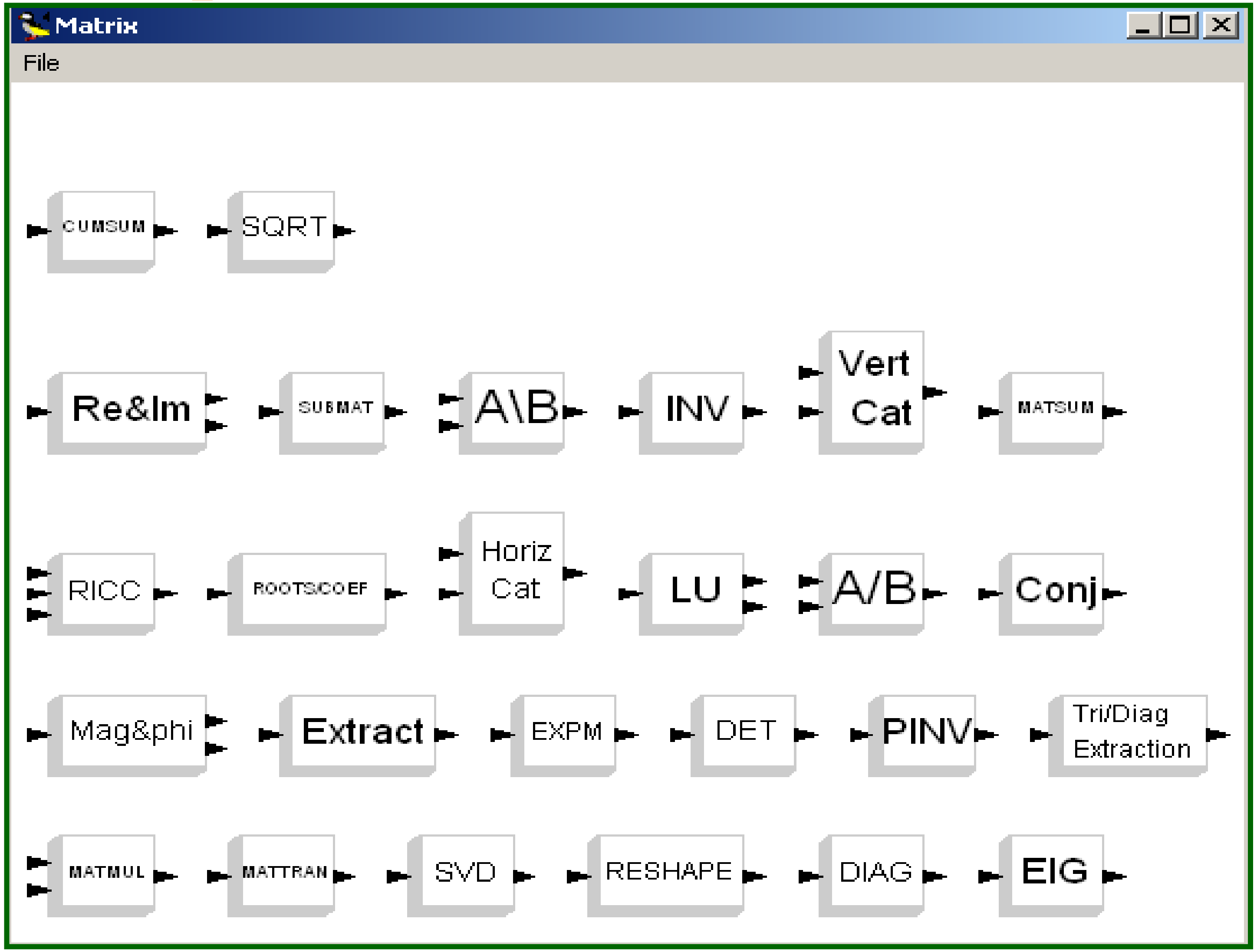
Output dati -- (menù Sinks)

Principali librerie Scicos (5)



Vecchi blocchi -- (menù OldBlocks)

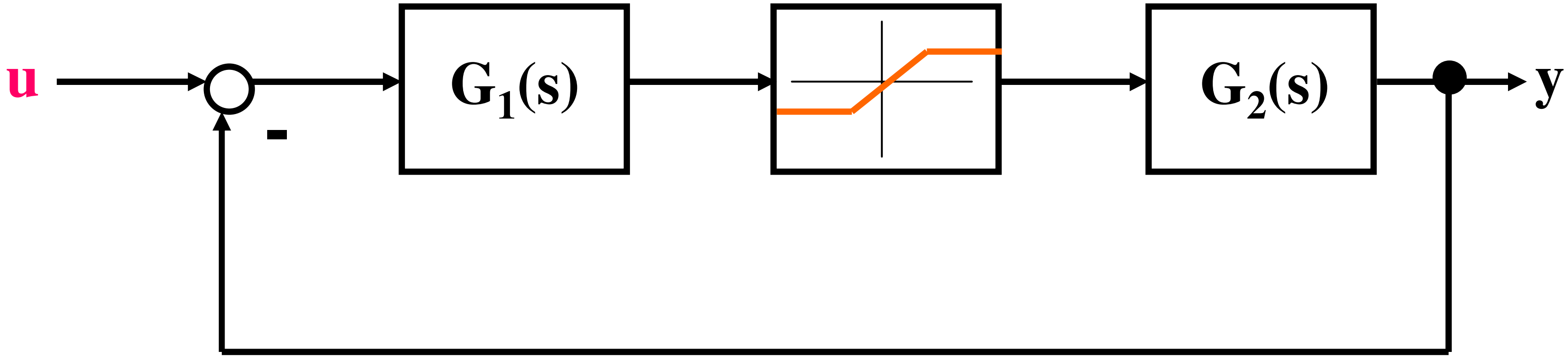
Principali librerie Scicos (6)



Algebra delle Matrici -- (menù Matrix)

Esempio

⌘ Si desidera **simulare** con **SCICOS** il seguente **sistema di controllo** che contiene una **NON LINEARITÀ**:



nel caso in cui il segnale d'ingresso sia il **gradino unitario**, cioè: **$u(t) = sca(t)$**.

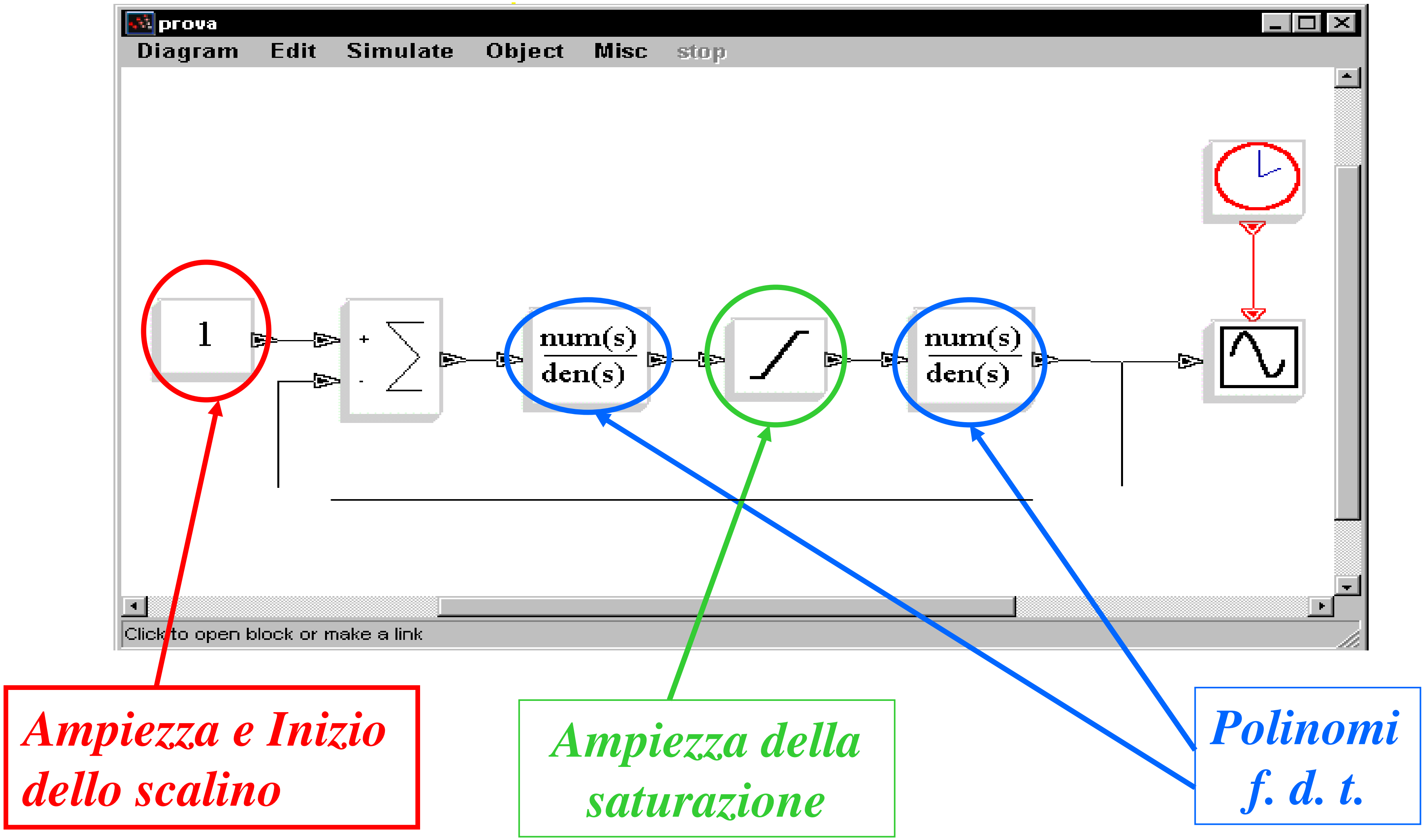
I vari BLOCCHI da utilizzare

- Blocco ***funzione di trasferimento***, menù ***Linear***
- Blocco ***saturazione***, menù ***Non-Linear***
- Blocco ***sommatore***, menù ***Linear***
- Blocco ***scalino***, menù ***Sources***
- Blocco ***grafico***, menù ***Sinks***

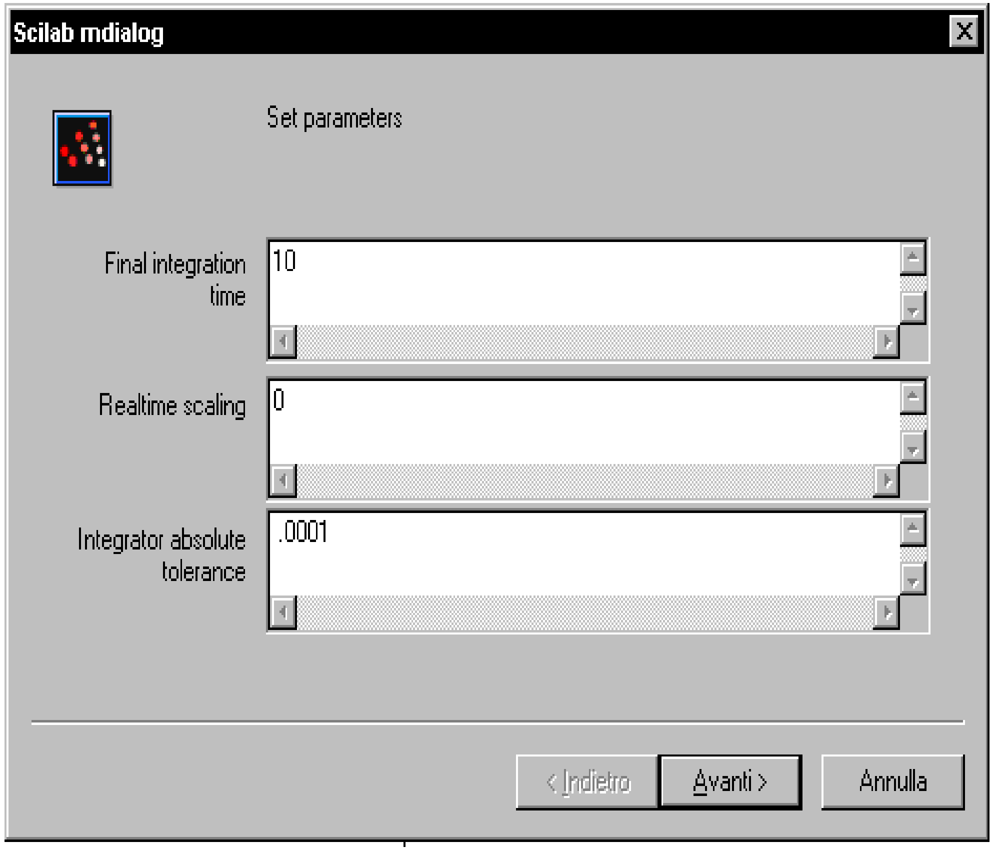
Le operazioni da eseguire sono:

- trascinare ciascuno dei blocchi nella finestra del modello;
- connetterli come nello schema a blocchi di partenza;
- definire i valori dei parametri di ciascun blocco.

Modello e parametri

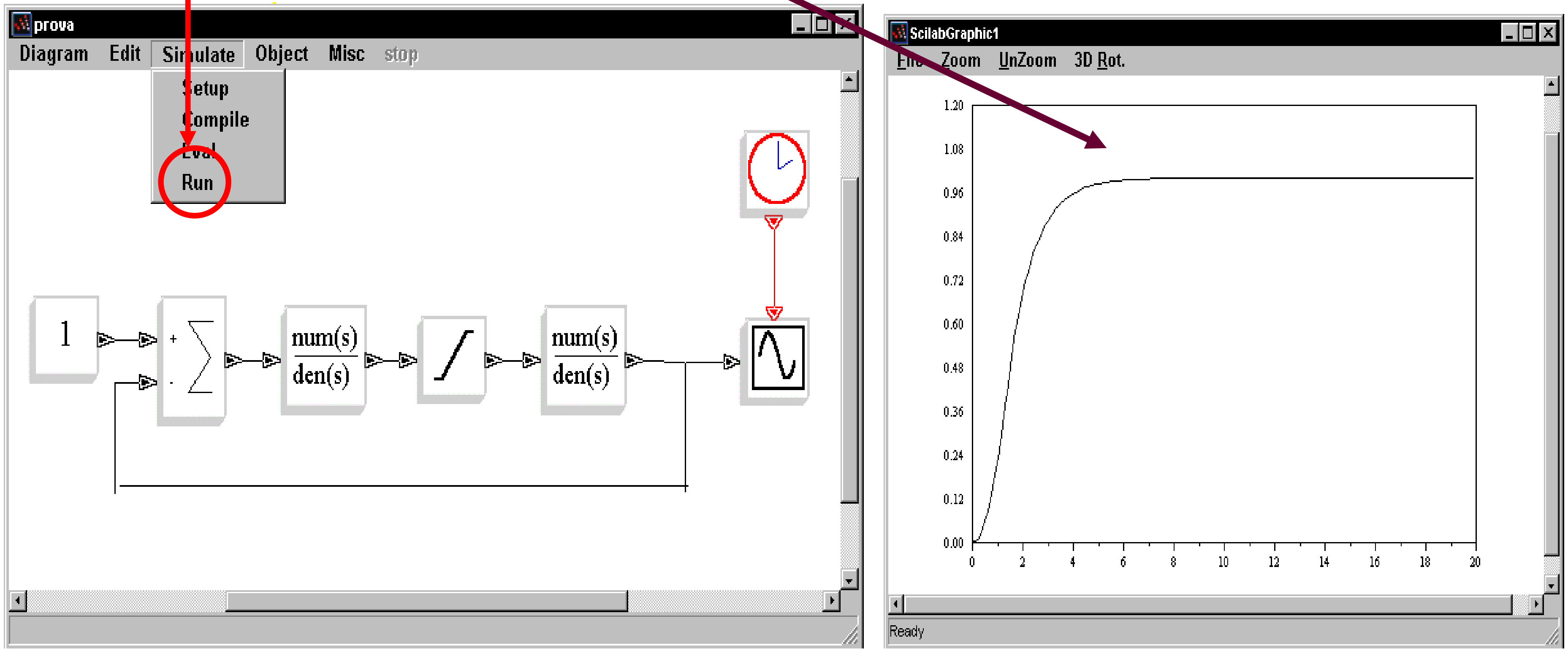


Parametri della simulazione



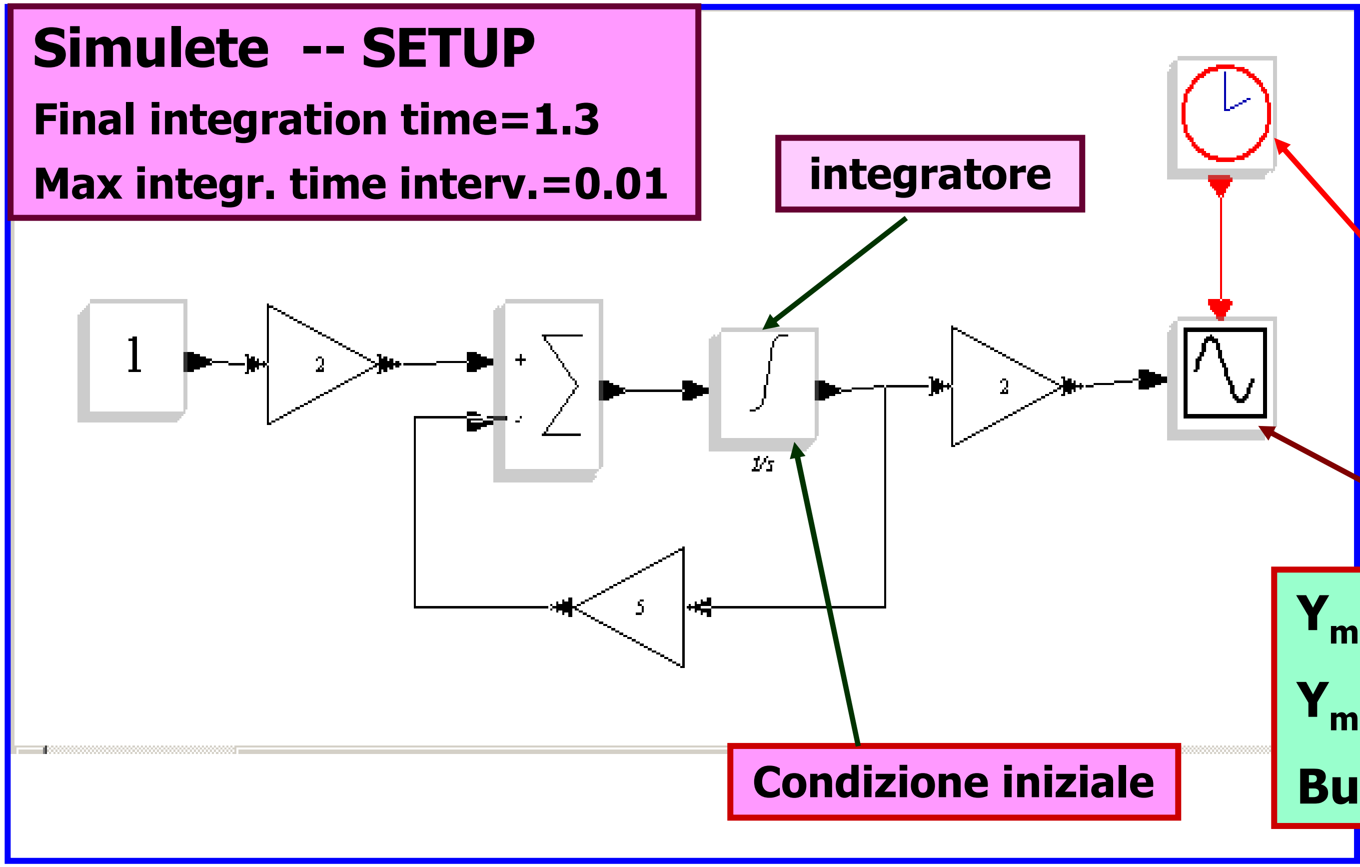
- L'utente deve definire:
  - l' **istante finale** della **simulazione**;
  - il **tipo** di **solutore numerico** (se il problema richiede metodi particolari);
  - i **parametri** del **solutore** (in genere, i default vanno bene).

AVVIO della SIMULAZIONE ed ANALISI dei RISULTATI



Esempio di simulazione
Spazio degli stati

$$\dot{x} = -5x + 2u$$
$$y = 2x$$
$$x_o = -1$$



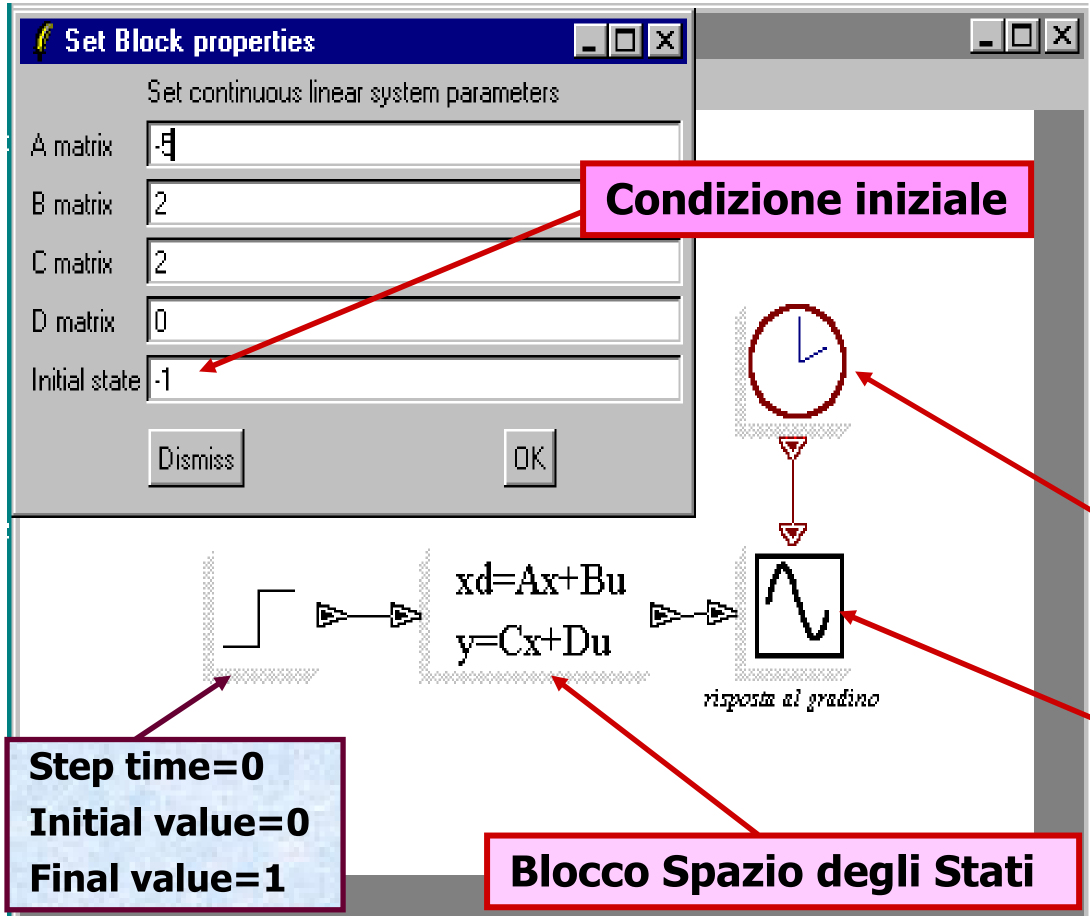
Y_{min}= -2

Y_{max}= 1

Buffer size = 2000

Esempio di simulazione
Spazio degli stati

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$



Condizione iniziale

Simulete -- SETUP
Final integration time=1.3
Max integr. time interv.=0.01

Period=0.01 Init time=0

Y_{min}= -2 Y_{max}= 1.3
Buffer size = 2000

Step time=0
Initial value=0
Final value=1

Blocco Spazio degli Stati

Risposta al gradino unitario

- Il **grafico** ottenuto dall'oscilloscopio presente nello schema a **blocchi** di **Scicos** viene mostrato nella figura seguente:

polo = -5 rad/sec

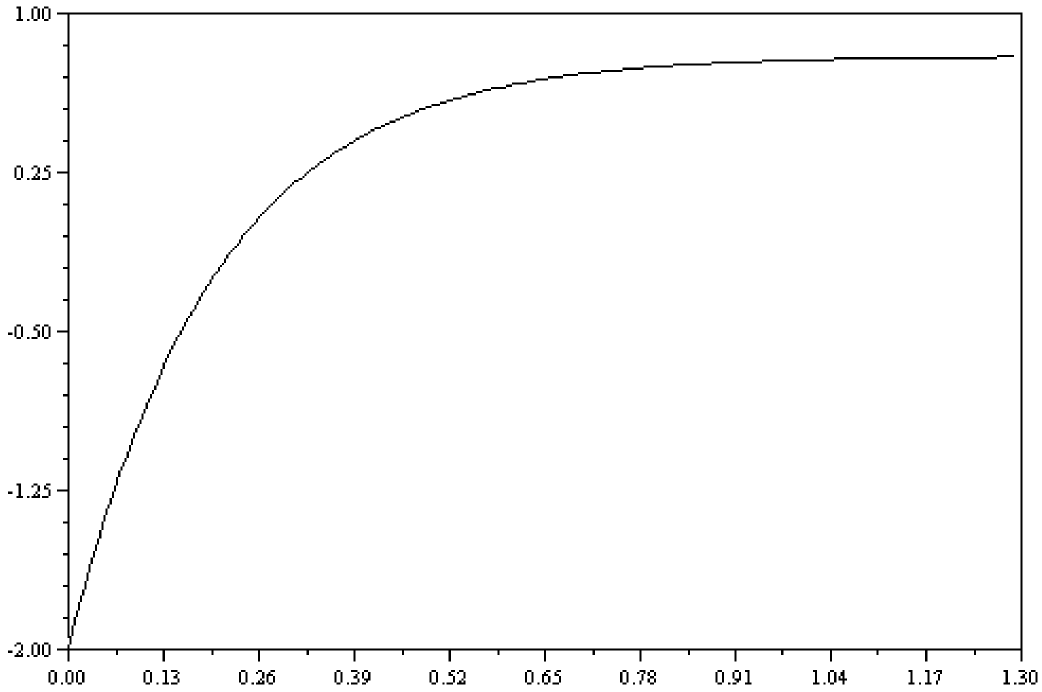
τ =| 1/p |=0,2 sec

y(∞) = 4/5 = 0,8

y(0) = 0

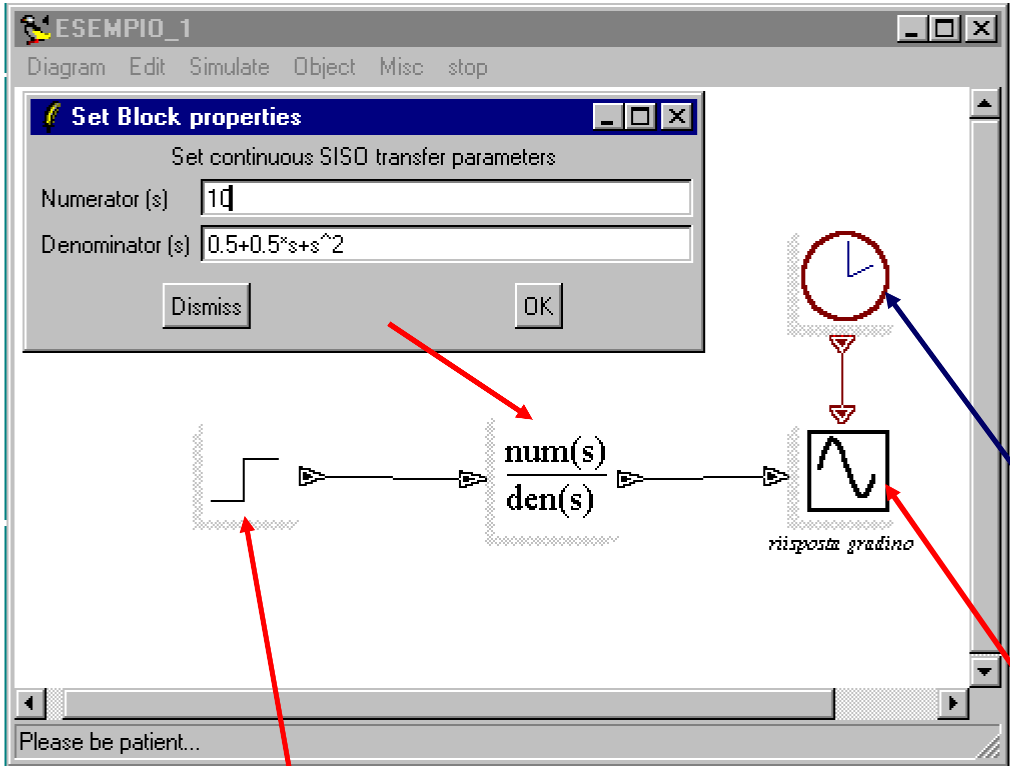
y'(0) = 4

$$\dot{x} = -5x + 2u$$
$$y = 2x$$



Esempio simulazione

$$G_1(s) = \frac{10}{s^2 + 0.5 * s + 0.5}$$



num=10;

den=poly([0.5 0.5 1],'s','c');

giesse1=syslin('c',num,den);

t=0:0.01:20;

y=csim('step',t'giesse1);

plot(t,y),xgrid

Period=0,1 Init time=0

Y_{min}=0 Y_{max}=28

Refresh period=20

Buffer size=10000

Step time = 0

Initial value = 0 Final value = 1

Risposta al gradino unitario

- Il **grafico** ottenuto dall'oscilloscopio presente nello schema a **blocchi** di **Scicos** viene mostrato nella figura seguente:

p_1=-0,25+0,661i

p_2=-0,25−0,661i

$T_a \cong 5/(\zeta\omega_n) \cong 20s$

y(∞) = 20

y(0) = 0

y'(0) = 0

$$G(s)=\frac{10}{s^2+0.5s+0.5}$$